Arabic Handwriting: Analysis and Synthesis

BY

Yousef S. I. Elarian

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

May 2014

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by Yousef S. I. Elarian under the direction of his dissertation advisor and approved by his dissertation committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING.

Dr. Abdelmalek Zidouri
(Advisor)

Dr. Wasfi G. Al-Khatib
(Co-Advisor)

Dr. Umar Al-Turki
College Dean

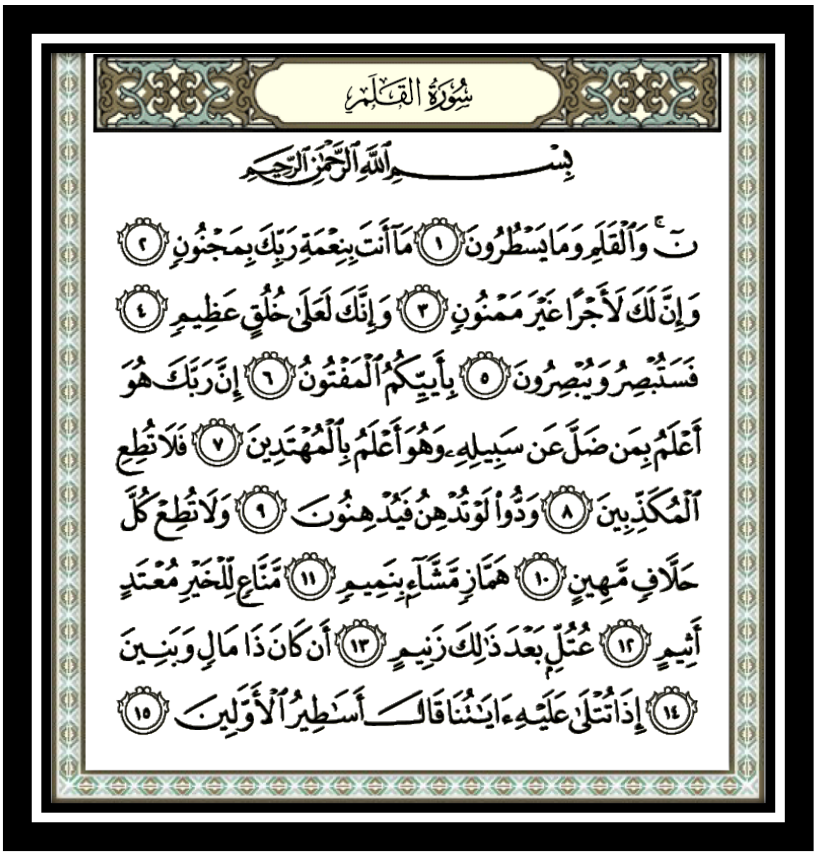
Dr. Moustafa Elshafei
(Member)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Radwan E. Abdel-Aal
(Member)

20/5/14
Date

Dr. Ashraf S. Hasan Mahmoud
(Member)



Dedication

لله

لوالدي

ثم لكل محب

To Allah

Then, to my parents,

and to all who helped, cared, or loved.

ACKNOWLEDGMENTS

Thanks again to my Lord, and to my Parents.

Thanks to King Abdul Aziz City for Science and Technology (KACST) for granting and supporting this work (Project # GSP-18-112).

Thanks to Dr. Sabri Mahmoud and to Dr. Muhammad Al-Mulhem.

Deep Personal Thanks to Dr. Zidouri, Dr. Al-Khatib, and the committee members.

Thanks to my colleagues: Sameh, Tanvir, Misbahuddin, Irfan, Anas,

and to all who helped that this dissertation is completed

Thanks from the heart.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	:	Two-Dimension
A	:	Isolated character-shape
ALSA	:	Adaptive Line Segmentation Algorithm for Arabic
B	:	Beginning character-shape
Bx	:	Extended beginning character-shape
BL	:	Baseline zone/range
BoH	:	Block of Handwriting
CAPTCHA	:	Completely Automatic Public Turing Test to Tell Computers and Humans Apart
CB	:	Character Bound
CC	:	Connected Component
CSC	:	Character-Shapes Covering algorithm
E	:	Ending Character-Shape
EG	:	Extended Glyph
GT	:	Ground-truth

GUI	:	Graphical User Interface
HL	:	High Ligativity
$HP^{(m)}$	:	Horizontal Projection (smoothed with factor ***m***)
IL	:	Isolated Characters
KW-PDF	:	Kashida Width Probability Density Function
LCD	:	Lower Contour Direction
LL	:	Low Ligativity
M	:	Middle character-shape
MBRE	:	Multiple Baseline-Range Estimation
OCR	:	Optical Character Recognition
OS	:	Over-Segmentation
PAW	:	Piece of Arabic Word
PB	:	Piece of Arabic Word Bound
PDF	:	Probability Density Function
PoD(s)	:	Part(s) of the Dataset
SE	:	Synthetic Extension
SBRE	:	Single Baseline-Range Estimation

UCD	:	Upper Contour Direction
US	:	Under-Segmentation
UT	:	Unligative Text
$VP^{(m)}$	:	Vertical Projection (smoothed with factor ***m***)
xE	:	Extended Ending Character-Shape
xMx	:	Extended Middle Character-Shape

ABSTRACT

Full Name : Yousef S. I. Elarian

Dissertation Title : Arabic Handwriting: Analysis and Synthesis

Major Field : Computer Science and Engineering

Date of Degree : 2014

Handwriting recognition and synthesis are challenging problems, especially for the Arabic script. However, synthesis, or the automatic generation of handwriting, has recently gained interest because of its various applications that include training recognition systems and font personalization.

In this dissertation, we addressed the problem of Arabic handwriting analysis and synthesis. We collected a dataset that is specifically designed for Arabic handwriting synthesis. This was accomplished by assuring that the dataset-text contains all Arabic characters and their shapes. Moreover, we introduced the idea of decoupling ligative from unligative texts to ease dealing with each separately.

The unligative dataset was and ground-truthed to the character-level, and an entropy-based measure was used to cross-validate the automatic and the semi-automatic results. The ground-truthed dataset and the adapted measure form a valuable resource for benchmarking segmentation systems.

The segmentation step produced two sets of character-shapes: strictly segmented character-shapes and extended character-shapes. The extended character-shapes were concatenated by setting the selected shapes in juxtaposition so that their extensions directly connect in what we call the extension-glyph technique. Strictly segmented character-shapes require synthetic extensions for their connection. Hence, these were modeled and generated for what we call the synthetic-extension technique.

We synthesized and recognized handwriting samples using the extension-glyph and the synthetic extensions techniques. Not only did the synthesized data improve the performance of a baseline recognition system on a popular testing benchmark, but also it

appeared natural to the eye. An improvement of 16.39% in the recognition performance of the baseline system was achieved when 8,652 synthetic extension samples were injected to the original training set of 2,322 words. An additional benefit of the synthesized data is that it can be regarded as an expanded ground-truthed dataset on its own.

ملخص الرسالة

الاسم الكامل: يوسف سالم عيسى العريان

عنوان الرسالة: تحليل وتصنيع خط اليد العربي

التخصص: علوم وهندسة الحاسوب

تاريخ الدرجة العلمية: 2014

إن التعرف على الكتابة اليدوية وتكثيفها لمن المسائل التي نواجه فيها تحديات علمية، لا سيما فيما يخص الكتابة العربية. وقد استقطب مجال التكثيف الآلي للكتابة الشبيهة باليدوية الاهتمام بسبب تطبيقاته العديدة، والتي منها تدريب المتعرفات الآلية وشخصنة خطوط الحاسوب.

هذه الرسالة متوجهة لتحليل وتكثيف خط اليد العربي، حيث قمنا بجمع قاعدة بيانتا صممت خصيصا لهذا الهدف، وذلك بجعلها مشتملة ليس على جميع أحرف اللغة العربية فحسب، بل وعلى جميع أشكال تلكم الحروف. كما قمنا بفصل ما يتراكب من الحروف مما لا يتراكب منها تمهيدا للتعامل مع كل فريق على حداً.

بعد جمع الفقرات الكتابية، قمنا بتقطيعها إلى أحرف بشكل شبه-آلي تارة، وبشكل آلي تارة أخرى. وقد قيمنا التقطيع الآلي بمقارنته بنظيره شبه-الآلي عبر مقياس خصصناه ليتلاءم مع هذه المهمة. وهنا نشير إلى أن البيانات المقعة بشكل شبه آلي تشكل –بشكل عام- مع المقياس المذكور أداة للتحكيم الكمي على التقطيع الآلي.

وقد نتجت عن عمليات التقطيع مجموعتان من أشكال الحروف: الحروف المقطوعة حدا والحروف المقطوعة بامتداد. فأم الممتدة، فتوصل بوضعنماذجها المختارة متجانبة حيث تلتقي الامتدادات فيما أطلقنا عليه أسلوب الحرف الممتد لتكثيف الكتابة. وأما الحروف المقطوعة حدا فإنها تحتاج لامتدادات صناعية تمت نمذجتها وتكثيفها واستخدامها فيما أطلقنا عليه أسلوب الوصلة الصناعية لتكثيف الكتابة.

وقد توجنا تكثيف الكتابة بتقديم نتاجها لمتعرف آلي من الظراز الأحدث، فازدادت نسبة تعرفه مقارنة بحاله عندما تدرب على قاعدة البيانات الأصلية فقط، وبلغت نسبة التحسن في المتعرف المزود بكتابة مكثفة بأسلوب الامتداد الصناعي 16.39%. هذا وتجدر الإشارة إلى أمرين: الأول أن الكتابة المكثفة امتازت بمشابهتها للحقيقية أمام العين، والثاني أن هذه النتائج تعد من النوع الذي يمكن تقطيعها لأصلها، مما يكسبها أهمية تقنية قصوى.

CHAPTER 1

INTRODUCTION

Handwriting is challenging, whether for analysis or synthesis, especially for languages that use the Arabic script. Analysis aims at gaining better understanding of a complex object by breaking it down into to smaller components [1]. Handwriting analysis usually encompasses segmenting handwritten images into characters.

Synthesis refers to a combination of two or more entities that together form something new; alternately, it refers to the creating of something by artificial means [2]. Synthesis of handwriting often aims at the automatic production of images that resemble, or perform like, those of human handwriting. Handwriting synthesis can be seen as the reverse operation of handwriting recognition: In recognition, handwritten images are given, and the corresponding text is output. In synthesis, a required text is given, and a corresponding handwritten-like image is output.

Synthesis has applications in the improvement of text recognition systems, in PC-personalization, in forgery detection, in Steganography (the art of hiding the existence of information), and in Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). These applications may require different specifications on the synthesized data, such as being of a specific writer's style or difficult to read by machines. Other characteristics of a handwriting synthesis systems include: whether the

data is online (with temporal information from tablets) or offline (on paper, without time stamps), the synthesis level (stroke, character, word, etc…), and the scripting system (Arabic, Chinese, Latin, etc…).

Handwriting synthesis may encompass *generation* [3]–[5] and *concatenation* [6]–[8] operations. Handwriting generation alters samples of handwriting to increase their shape-variability within some closed-vocabulary. Concatenation operations, in contrast, aim at the compilation of new units of vocabulary, such as words, from a smaller pool of basic samples, such as characters. Handwriting generation can be seen as the inverse operation of preprocessing in a text recognition system whereas handwriting concatenation can be seen as the inverse operation of segmentation.

1.1 Problem Statement

Handwriting recognition requires training samples that capture as much as possible of the natural variability of handwriting styles [9]. Moreover, it requires the samples to contain *ground-truth* (GT) information that aligns the underlying text with the corresponding images at some level. The conventional ways of collection and ground-truthing encompass manual tasks that can be very laborious and time-consuming. Hence, researchers have proposed the use of synthesized data in expanding training sets of recognition systems [10]–[14].

The insertion of synthesized data in a training set can have benefits and side effects. While the increased variability of the training set may lead to the recognition of otherwise mal-recognized examples, distorted samples may disturb the parameters of a

recognition system from their adequate values. The overall impact of any proposed method needs to be positive in terms of recognition rates. Intuitively, we can expect that *naturally looking* data are more promising to avoid distorting the parameters of a recognition system while improving its recognition performance. Hence, we aim at the synthesis of Arabic handwriting that looks natural and improves the accuracy of recognition systems.

Concatenation-based systems can provide a means of open-vocabulary synthesis. However, concatenation calls for character-segmentation, a quite challenging problem, especially for the Arabic script. One main cause for the lag in solving Arabic segmentation is the severe lack of appropriate ground-truthed datasets for its benchmarking. Since ground-truths, themselves, consist of labeled segmented handwriting, ground-truthing and segmentation engage in a "chicken and egg" relationship: the ground-truth data is needed for the development and evaluation of segmentation systems, and segmentation systems are needed to speed ground-truthing up.

One way to break this recursion is by implementing text-aware alignment systems. These can result in accurately labeled (segmented) data for the special circumstances where the text is known, like in certain datasets. Another way out is to find subjective and objective semi-automatic alternatives for ground-truths for segmentation evaluation. For all of the above, it is useful to expand small amounts of manually ground-truthed data via handwriting synthesis

1.2 Motivation and Applications

Researchers cite the lack of datasets of Arabic handwriting as a reason for the lagging-behind in Arabic writing recognition. Conventional ways of collecting datasets directly from writers have some disadvantages:

- Collection is costly in terms of time and effort.
- Once a dataset is designed and collected, adding new words to it can be difficult.
- Ground-truthing usually necessitates human interaction; hence, it is time-consuming.

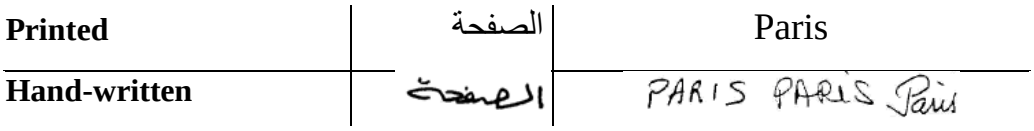
Synthesized data can improve systems that have deficiencies in their text segmentation accuracy, recognition features and classifiers, or variability of training data. In practice, all these are not perfect and may benefit from the use of synthesized data to improve recognition rates. Hence, one motivation to synthesize data is to expand text recognition training sets independently from their underlying recognition system [10], [12]. Other applications that demand handwriting synthesis include:

- Word spotting [15] and holistic recognition [16]
- Writer imitation/authentication [17],
- Personalized fonts generation [4],
- CAPTCHA generation [18], and
- Aesthetical calligraphy generation [19].

1.3 Background and Pre-Analysis of the Arabic Writing System

As a native language, Arabic is used by more than 200 million people around the world [20]. In addition, there are around 1.6 billion Muslims with some association to Arabic due to religious reasons [21]. The Arabic alphabet is also used to write Jawi, Urdu, Persian and other languages.

In Arabic, most characters obligatorily connect to their within-word successors. The Arabic character Hamza "ء" does not connect to either its precedent or to its successor, even if in the same word. Six other Arabic characters ("ا", "د", "ذ", "ر", "ز", and "و") and some Hamza-diacritized variants of them, never connect to their successors in the same word. These characters cause words to separate into unconnected pieces of Arabic words (PAWs). Spaces between PAWs are typically smaller than inter-word spaces. Figure 1 shows samples of printed and handwritten texts for Arabic and Latin scripts.



(a) **(b)**

Figure 1. Printed and handwritten samples for Arabic (a) and Latin (b) scripts.

Arabic characters can take up to four shapes depending on their positions in a PAW. From right to left (the Arabic writing direction), the first character in an Arabic PAW takes a character-shape that is called the beginning shape (B). A (B) shape in a

PAW can be followed by one or more middle shaped characters (M) before an ending shaped character (E) ends it. If a PAW consists solely of one character, it takes a shape called the isolated shape (A). In regular expressions, Arabic PAWs are expressed as <(A) | (B)(M)*(E)>, where the bar symbol "|" denotes the "OR" operator, and the star symbol, "*", denotes zero or more occurrences of the character-shape it follows. Figure 2 shows a word divided into three PAWs: PAW1 consisting of an (A) character-shape, PAW2 consisting of a (B) and an (E) character-shapes and PAW3 consisting of a (B), an (M), and an (E) character-shape. PAW1, PAW2 and PAW3 of Figure 2 are examples for the <(A)>, <(B)(E)> and <(B)(M)(E)> expressions, respectively. The PAWs in the figure are ordered from right to left, as this is the direction of Arabic script.

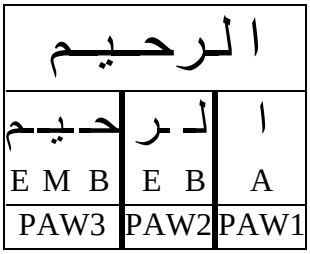


Figure 2. An Arabic word with three PAWs.

Table 1 shows a list of the 29 Arabic characters along with extra Arabic-used keyboard characters [22], [23]. The number of the character-shapes is 117.

Table 1. A list of Arabic characters and their different shapes based on their positions within PAWs.

Character Names	Alone (A)	Ending (E)	Middle (M)	Beginning (B)
Hamza	ء			
Alef with Madda Above	آ	آ		
Alef with Hamza Above	أ	أ		
Waw with Hamza Above	ؤ	ؤ		
Alef with Hamza Below	إ	إ		
Yeh with Hamza Above	ئ	ئ	ئ	ئ
Alef	ا	ا		
Beh	ب	ب	ب	ب
Teh Marbuta	ة	ة		
The	ت	ت	ت	ت
Theh	ث	ث	ث	ث
Jeem	ج	ج	ج	ج
Hah	ح	ح	ح	ح
Khah	خ	خ	خ	خ
Dal	د	د		
Thal	ذ	ذ		
Reh	ر	ر		
Zain	ز	ز		
Seen	س	س	س	س
Sheen	ش	ش	ش	ش
Sad	ص	ص	ص	ص
Dad	ض	ض	ض	ض
Tah	ط	ط	ط	ط
Zah	ظ	ظ	ظ	ظ
Ain	ع	ع	ع	ع
Ghain	غ	غ	غ	غ
Feh	ف	ف	ف	ف
Qaf	ق	ق	ق	ق
Kaf	ك	ك	ك	ك
Lam	ل	ل	ل	ل
Meem	م	م	م	م
Noon	ن	ن	ن	ن
Heh	ه	ه	ه	ه
Waw	و	و		
Alef Maksura	ى	ى		
Yeh	ي	ي	ي	ي

Arabic characters usually connect horizontally within an imaginary line that we refer to as the baseline (BL). The simplest and most frequent form of connecting

consecutive Arabic characters is through a semi-horizontal stroke called the Kashida. The Kashida stroke, shown in Figure 3, can vary in length, shape and thickness depending on the writing style. The figure also shows vertical overlapping between several characters and PAWs in the handwritten sample, which is a common case in the Arabic script. It also shows a broken character (the rightmost ـ character).

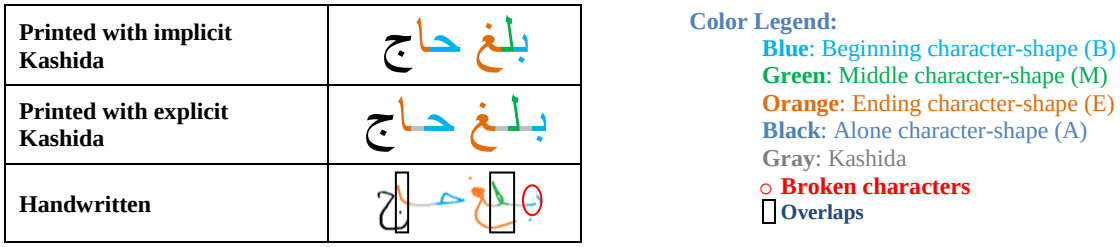


Figure 3. Arabic printed and handwritten samples colored to distinguish their (B), (M), (E) and (A) character-shapes.

We define l*igatures* [24] as alternate forms that replace certain sequences of characters in a way that is deformed from their direct concatenation. We use the terms *ligative* or *ligaturisable* for sequences of two or more characters that accept to be connected with a ligature. We use the term *unligative* or *ligatures-free* for sequences of two or more characters that only accept to be connected with a simple extension on the baseline. Ligatures are mainly used for aesthetic reasons. They can also play a role in making a writing more compact [25]. Figure 4 shows characters that are ligated in one instance of a Qur'an script and unligated in another [26].

(a) **(b)**

Figure 4. Examples of the same script from the Holy Qur'an (a) without and (b) with a ligature.

Calligraphic conventions determine which connectable character sequences are ligative. Except for the *Lam-Alef* family (ﻻ،ﻷ،ﻹ،ﻵ), where ligation is obligatory, actual ligation is a writer's choice. In other words, being ligative is a necessary but not sufficient condition for ligation (forming a ligature).

The frequency of ligature usage in a document may depend on the font or handwriting style, the level of formality of the document content (*e.g.* poetry *vs*. business documents) and on other factors [27]. In general, the frequency of ligatures in handwritten documents tends to exceed their frequency in modern printed documents.

Figure 5 shows three pairs of connectable characters, hereafter referred to as *bigrams*. Figure 5(a) is an example of a ligative bigram that can be optionally written as a ligature. Figure 5(b) shows an instance of an unligative bigram since it does not encompass any ligative form. Figure 5(c) shows an instance of the obligatorily ligative family of bigrams

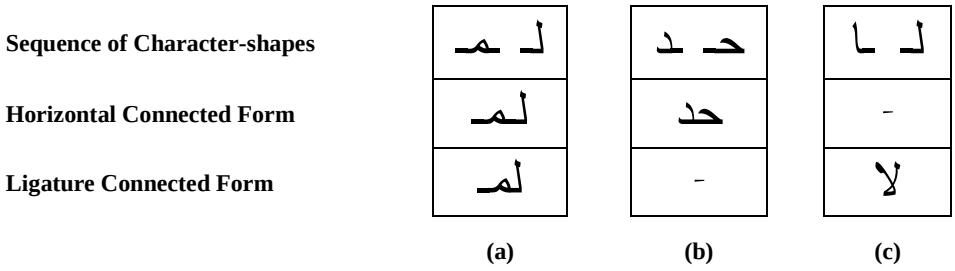


Figure 5. Examples of (a) a ligative, (b) an unligative and (c) an obligatorily-ligative characters.

Arabic characters may have ascenders that go above the BL range (as exemplified in the dashed box of Figure 6), descenders that go below it (exemplified in the dotted box of Figure 6), or curvy shapes within the BL zone (like the area surrounded by the dash-dotted diamond in Figure 6). Arabic characters may have secondary diacritics and dots

above or below the primary glyph of a character (surrounded by ellipses in Figure 6). Arabic characters vary considerably in width and height (see Appendices A and B for statistics on these).

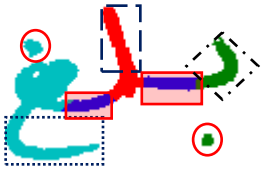


Figure 6. A word with Kashida (solid boxes), ascender (dashed box), descender (dotted box) and dots (ellipses)

1.4 List of Contributions

In this section, we list the most important contributions of this dissertation along with pointers to the most related chapter or section in the dissertation (enclosed between parentheses):

- We wrote the first and only standalone survey document on shape-simulation handwriting synthesis. It includes original classifications of techniques, applications, and evaluation methods and a literature survey for each of these classifications (Chapter 2).
- Statistics on the widths, heights, and character frequencies of the Arabic language and script (Section 4.4.2, and the Appendices A to E).
- We designed, collected and digitized a dataset of Arabic handwriting (Chapter 3). In particular, a list of Arabic ligatures and a paragraph of non-ligaturisable text are designed with compactness and comprehensiveness analysis.
- We have developed, implemented and tested segmentation algorithms on the line, word, PAW, and character levels (Chapter 4).

- Representative samples of Arabic handwriting have been ground-truthed to the pixel-level, and an entropy-based measure was adapted for the evaluation of Arabic handwriting segmentation against the aforementioned ground-truth (Chapter 4).
- We have developed algorithms and tools for synthesis and used them to synthesize masses of pixel-level ground-truthed data for two different scenarios (Chapter 5).
- A statistical framework for generating simple writing strokes from their width and direction parameters is demonstrated (Section 5.4).
- The impact of synthesized data is investigated on a recognition system, and promising results are reported and compared (Chapter 5).

1.5 List of Publications

This work has resulted in several journal and conference papers, some of which are still under review. The following list includes our journal papers.

- **Yousef Elarian**, Irfan Ahmed, Sameh Awaida, Wasfi Al-Khatib and Abdelmalek Zidouri, *An Arabic Handwriting Synthesis System*, Pattern Recognition, (submitted for revision).
- **Yousef Elarian,** Radwan Abdel-Aal, Irfan Ahmed, Tanvir Parvez, & Abdelmalek Zidouri, *Handwriting Synthesis: Classifications and Techniques*, the Inter. Journal of Document Analysis and Recognition (Accepted).
- **Yousef Elarian**, Wasfi Al-Khatib, Sameh Awaida, and Abdelmalek Zidouri. *Ligatures in the Design of Comprehensive Arabic Datasets*, the Journal of Language Resources and Evaluation (submitted for revision).

- **Yousef Elarian**, Wasfi Al-Khatib and Abdelmalek Zidouri, *A Segmentation Benchmark for Arabic Handwriting Based on a Character-Level Ground-Truth and a Novel Evaluation Metric,* the Journal of Language Resources and Evaluation (submitted for revision).

The following list includes the conference papers that are related to this work.

- **Yousef Elarian**, Abdelmalek Zidouri, Sameh Awaida, and Wasfi Al-Khatib, "Ligatures in the Design of Arabic Handwritten Datasets," Document Engineering 2014, Fort Collins, Colorado (submitted for revision).
- **Yousef Elarian**, Abdelmalek Zidouri, and Wasfi Al-Khatib, "Ground-truth and Metric for the Evaluation of Arabic Handwritten Character Segmentation," the 14th International Conference on Frontiers in Handwriting Recognition ICFHR2014.
- **Yousef Elarian**, Husni Al-Muhtaseb and Lahouari Ghouti, *Arabic Handwriting Synthesis*, in *International Workshop on Frontiers in Arabic Handwriting Recognition*, Istanbul, 2011.
- **Yousef Elarian** and Sabri A. Mahmoud, "An Adaptive Line Segmentation Algorithm (ALSA) for Arabic," Inter. Conference on Image Processing, Computer Vision, and Pattern Recognition, pp. 735–9, Las Vegas, 2008.
- **Yousef Elarian,** "Arabic Handwriting Synthesis," 1st Saudi Higher Education Students Conference, Riyadh, 2010 (winner of the 4th conference prize).
- **Yousef Elarian**, Sameh Awaida and Sabri Mahmoud. *"*Design of Datasets for Handwritten Arabic Texts Research*,"* 1st Saudi Higher Education Students Conference, Riyadh, 2010.

- **Yousef Elarian**, "Analysis of Some Arabic Scripting Units in Computational-Linguistic Resources," 1st Saudi Higher Education Students Conference, Riyadh, 2010.

1.6 Dissertation Organization

Typical concatenation-based synthesis systems for cursive writing involve a segmentation phase and a concatenation phase. The sequence of the remaining chapters in Figure 7.

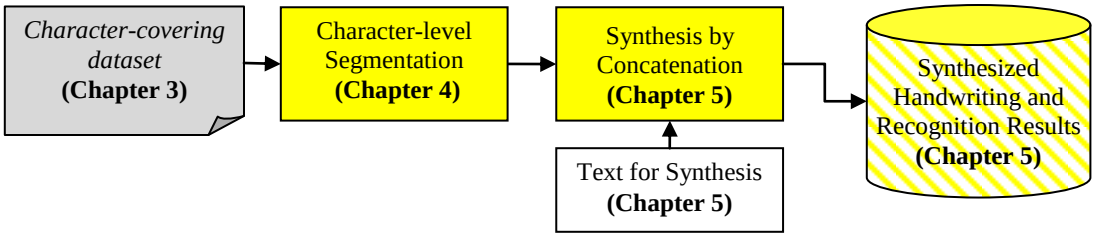


Figure 7. Blocks of a concatenation-based synthesis system and their main discussion sections.

The rest of this dissertation is organized as follows: Chapter 2 presents our classifications and surveys of synthesis in literature. Chapter 3 presents analysis of Arabic handwriting and discusses the design of our dataset and the steps towards extracting handwriting samples from it. Chapter 4 documents our processes and results of segmenting and aligning blocks of handwritings into ground-truthed character-shapes and opens doors for pixel-level analysis. Chapter 5 presents our synthesis approach and results. Finally, we conclude and recommend future work in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

Handwriting synthesis refers to the artificial generation of data that resembles human writing. Synthesis has applications such as the improvement of text recognition systems, PC-personalization, calligraphic fonts, forgery detection, and Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). These applications may require certain specifications on the synthesized data, such as being of a specific writer's style or a specific script. Applications also suggest methods to evaluate the adequacy of synthesized data.

Handwriting synthesis can model handwriting either via the simulation of the human writing process (*top-down* approach) or via the mere imitation of its outcome (*bottom-up* approach). In the top-down approach, the neuromuscular acts of writing are simulated in what is commonly termed as *movement-simulation*. When the data itself is regenerated without imitating human movements, synthesis is termed as *shape-simulation* [28].

Some synthesis systems can be seen as the reverse of more well-known applications. For example, when synthesis aims at the generation of individual characters from their ASCII codes, it can be regarded as the reverse of character recognition. Similarly, when synthesis aims at the generation of words through the concatenation of characters, it can be regarded as the inverse of character segmentation.

Handwriting synthesis is a hot topic with increasing interest from the research community. Among the refereed journals that contribute to the dissemination of established knowledge in the area are: the International Journal of Document Analysis and Recognition (IJDAR) (*e.g.* [14], [29], [30]), Pattern Recognition (*e.g.* [18], [31]–[33]), Pattern Recognition Letters (*e.g.* [34]), Machine Learning (*e.g.* [35]), and others. Besides, some prestigious conferences such as the International Conference on Document Analysis and Recognition (ICDAR) (*e.g.* [36]–[39]), the International Workshop on Document Analysis Systems (DAS) (*e.g.* [12]), the International Conference on Pattern Recognition (ICPR) (*e.g.* [40]–[43]), and the International Conference on Frontiers in Handwriting Recognition (ICFHR) (*e.g.* [4], [44]–[46]) help in spreading the advances in the field. In this chapter, we survey synthesis techniques with focus on shape-simulation approaches.

2.1 Synthesis Applications, Specifications and Evaluation Methods

The applications of synthesis guide the specifications (requirements and constrains) of synthesized data and suggest methods to evaluate the corresponding synthesis systems. In this section, we identify some handwriting synthesis applications and link them to the specifications and evaluation methods that may suit them.

2.1.1 Synthesis Applications

Handwriting synthesis has a wide range of applications. It can be used to generate desired and inexpensive ground-truth data for the development of text segmentation and recognition systems [47]. A recent application of synthesis is CAPTCHA. Synthesis can

also be a means for fonts personalization [48], [49]. Synthesis with writer-imitation can be used for calligraphy generation, word spotting, and writer identification.

Synthesized handwriting might target humans, machines or both. It may be intended to imitate a particular writer's style, to generate writer-independent handwriting, or to tell humans and machines apart. Synthesized calligraphy, for example, targets human subjects [17], [44], [50] while generic training data targets text recognition systems [11], [12], [34]. Then again, word spotting systems may benefit from writer-specific synthesis to find words written by a particular scribe [15], [51] and from generic synthesis to find words regardless of scribes. Some synthesis applications may require human legibility but low machine readability [52].

Figure 8 shows some applications on a Machine-Human readability plane. CAPTCHA is a test used to ensure that a response is generated by a human, not a computer. Handwritten CAPTCHAs, in particular, exploit the gap between humans and machines in reading handwriting [18]. Similarly, calligraphic and personalized fonts aim at the aesthetic aspects of writing but may be confusing to machines. On the other hand, some perturbed and noisy text which might not be pleasant to humans can be useful for training recognition systems [11], [38], [41]. Steganography, the art of hiding data, is another application for synthesized handwriting where secret messages can be communicated by certain choices of the optional features in a script [53], [54].

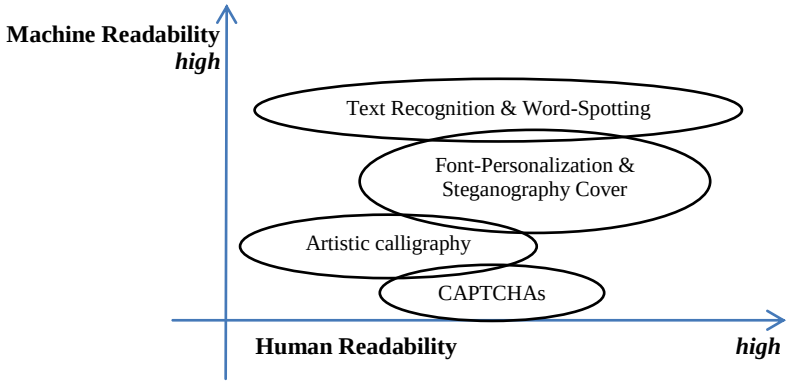


Figure 8. Applications of handwriting synthesis on the Human- vs. Machine-Readability graph.

2.1.2 Specifications of Synthesis Systems and Outputs

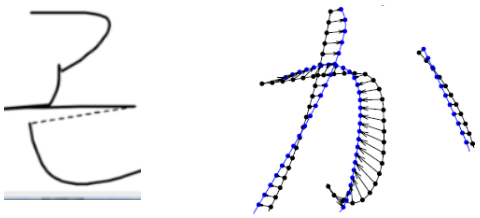
There are several aspects of the synthesized data that can be specified based on their application. One, or occasionally more, specifications for each of the following aspects can be used to describe a synthesis system:

- **Input/output levels relationship:** Generation *vs*. concatenation system
- **Output level:** Stroke, character, character group/PAW, word, line or paragraph
- **Data types:** Online *vs.* offline
- **Writing script:** Arabic, Chinese, Indian, Latin, etc…
- **Parameterization:** parametric *vs.* non-parametric system
- **Writer-imitation:** Writer-specific vs. writer-independent

The input/output levels relationship and the parameterization aspects specify synthesis systems, rather than their outputs. The data types' aspect may specify input or output data. The rest of the aspects strictly describe specifications of the outputs of synthesis systems. The first two aspects are discussed jointly while the remaining ones are discussed in the subsequent subsections.

Input / Output Levels

Handwriting synthesis receives images of handwritten samples and generates output handwriting images. The input and output images can be at different levels of writing units such as sub-characters, characters, words, lines, or paragraphs. Based on the relationship between the levels of the input units and the output units, we classify synthesis techniques into: generation techniques and concatenation techniques. Generation techniques produce new synthesized images at the same level of the input samples they receive. Concatenation techniques, in contrast, produce output images at higher levels than their inputs. Figure 9 and Figure 10 show examples of generation and concatenation synthesis, respectively. The levels of the output units in Figure 9(a) [55], Figure 9(b) [19] and Figure 10(a) [3] are characters while Figure 10(b) [18], (c) [56], and (d) [4] correspond to words, lines and paragraphs, respectively.



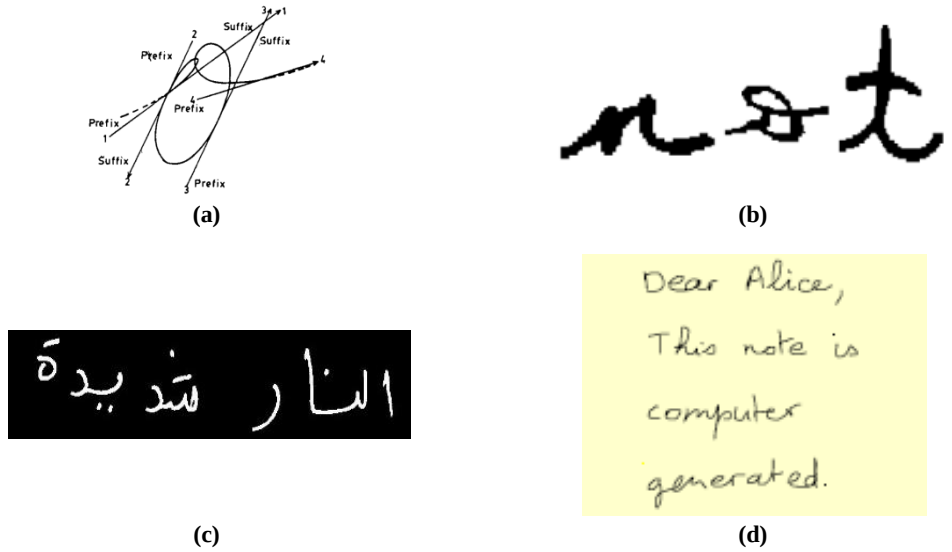
(a) **(b)**

Figure 9. Generation of (a) Hangul and (b) Hiragana character.

Data Types

Online data, such as coordinate time-stamps and pressure, are captured as writing occurs on special devices called tablets. Offline data are taken as static images of script

that are written on paper. Figure 9(a) and Figure 10(a) show online data. Offline data lacks temporal information but contains inking and stroke-thickness information (*e.g.* Figure 10(b)). Usually, the data types of the inputs and the outputs of synthesis systems are the same. Sometimes, however, online data might be used to generate offline-like outputs, often by the addition of inking effects [4], [12], [50]. In addition, some systems utilize a mixture of online and offline data in their inputs (*e.g.* Figure 9(b)) such as when a printed character is used as a standard reference for handwritten samples [57].



(a) **(b)**

(c) **(d)**

Figure 10. Concatenation for (a) Latin online character from sub-characters (b) Latin offline word from characters (c) Arabic line from characters and (d) Latin paragraphs from PAWs.

Writing Scripts

A script can be used to write several languages. The Latin script, for example, is used in English and Spanish languages. A script can be inherently cursive as in Arabic, inherently discrete as in Hiragana [58] and Katakana [59], or mixed as in modern Latin. Researchers have worked on synthesis of Latin [4], [44], Arabic [14], [56], Cyrillic [34],

Chinese [36], [60], Korean (Hangul) [12], Japanese ([19], [58] and [59]) and Indian (Hindi, Tamil, Malayalam, and Telugu) scripts [46]. Occasionally, systems are implemented and tested on multi-scripts [15], [29], [34].

Parameterization

The number of parameters a synthesis technique involves is an important aspect to study. In general the less the number of parameters the preferable it is. But sometimes, more parameters provide increased flexibility in deciding the desired quality of synthesized text. Parameters may also affect the computational efficiency of a technique. Another important aspect of parameters is their estimation/training. Some techniques may involve parameters which require expert knowledge for calibration while other parameters may be trained from the data available. Moreover the number of parameters that need to be trained also places some constraint on the minimum data required to robustly train the model [61].

Synthesis systems may differ in the ways how they are parameterizable [62]. Parametric models use observable parameters to define a system. Non-parametric models, *e.g.* statistical models, may still use parameters; but these usually lack physical meaning [63]. Sigma lognormal models [40], [64], as well as signal-based models [65] and spline-based models [3], [44], depend on parameters for the definition of character-shapes. Parameterization may be used to smooth joining ligatures between characters in concatenation systems [18], [48]. In generative systems, changes to samples are controlled via parameters. For example, perturbation is added to samples, as in [6], [11],

[39]. *Naturalness* can be parameterized, as in [66], where the relative distance from the printed sample and the nearness to handwritten sample is considered *naturalness*.

Writer-Imitation

Synthesis may or may not aim at the imitation of a specific writer's style, depending on their applications. Synthesis for character recognition improvement [3], [5], [12], [58], as well as for CAPTCHA generation, usually lacks writer-specific features [18], [67]. On the other side, applications such as PC-personalization [4], [17], [44], [66] and writer-identification [17], [68], [69] call for writer-specific synthesis. In Table 2, we classify the applications of handwriting synthesis by their writer-imitation and target aspects. In some cases (*e.g.* [40]), large databases of handwriting can be synthesized to generate writing samples for a single writer as well as in multi-writer setup. Some researchers [55] [45] develop systems that can function in either a writer-independent or a writer-specific modes.

Table 2. Reported works on handwriting synthesis for the human and machine targets.

Writer-Imitation Target	**Writer-Independent**	**Writer-Specific**
Human	Pen-Based PC [28], [44] Calligraphy Arts [17], [50] CAPTCHAs [18], [67]	Writer-imitation [56], [65] PC- Personalization [4], [17], [45], [50], [66], [70]
Machine	Text Recognition [3], [5], [11], [11]–[13], [17], [37], [39], [41], [42], [56], [58], [65], [71]–[73] Word-Spotting [56] Compression [3], [34]	Writer Identification [17], [39], [65], [69] Word-Spotting [56]

2.1.3 Evaluation Methods

The choice of evaluation methods for synthesized data depends on the application domains for which the synthesis system is designed. Commonly used evaluation methods fall into two main categories: subjective and objective.

S*ubjective* evaluation methods mainly rely on the opinion of human subjects. In few cases, trained subjects may decide if some handwriting belongs to a specific writer. Several researchers have used subjective methods for evaluating the synthesized handwriting. Subjective opinions of 21 English native speakers, that were not among the 15 writers of the database of [73], were used to evaluate the performance of their parameter calibration. For example, Guyon mentioned that in subjective evaluation, the trained eye can find exaggerated regularities in character-shapes and probable inconsistencies in inking [4]. Other works that rely on subjective evaluation include [17], [44], [55].

Objective methods rely on quantitative measures for the evaluation of synthesized handwriting. Text and writer recognition systems give success rates which can be used as measures of the machine-readability or writer-resemblance of some handwriting [10], [69]. In order to assess data that is synthesized for OCR improvements, the data can be injected to the training set. Injecting more synthesized data to training data is expected to improve the performance of the recognizer under the condition that the synthesized data captures variability of natural writing. The premise is taken from a rule of thumb with real data: the more training data the better the recognition [74].

Figure 11 shows the most common evaluation methods grouped into the subjective and objective criteria.

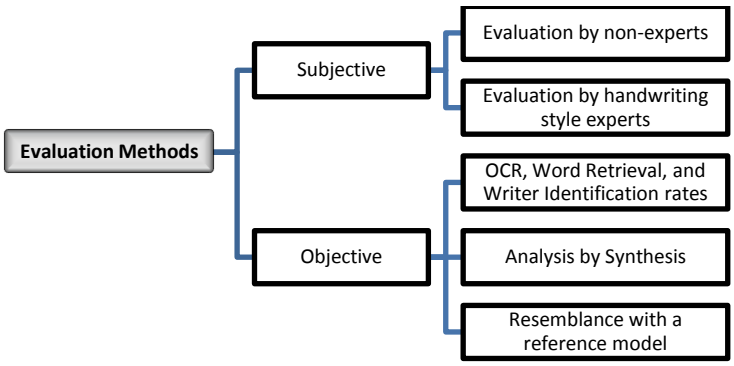


Figure 11. Most common methods to evaluate synthesized data.

Improvements in HMM-OCR performance on the IAM database were reported after the injection of synthetic training data in [38] and [10]. Support vector machine OCR that runs on a database of 10 Hiragana characters (from the HANDS-nakayosi t-98-09 database) was used in [12], with reported improvements on the OCR performance. Similar efforts for improving OCRs using synthesized data include [5], [69], [71], [75]. A script recognizer was used to classify synthesized text into Arabic, Latin or Russian by Vincent *et al.* [34]. Although all of their synthesized data was perfectly labeled with its correct script type, the authors commented that the differences between correlation coefficients were quite small and not very reliable. In [37], normal OCR Turing test is used for the evaluation of synthesized Arabic handwritten. The models derived in [65] achieve 99.4% success rate when tested as recognizers.

Analysis by synthesis is an objective evaluation method that judges synthesizers by the quality of their recognition models. This evaluation method is especially useful with generative model-based synthesizers. An analysis by synthesis scenario was used in

[65]. They performed a test of completeness on their statistical model to demonstrate the ability to recognize data not in the training set.

Another objective evaluation method for synthesis compares synthesized handwriting to some *reference model*. Dolinsky and Takagi consider printed Hiragana characters as reference models that are deformed by personal handwriting styles [76]. Correlations and regression analysis are used to quantify the difference between the synthesized and reference model. Zheng *et al.* [39] also quantify the amount of deformation needed for their fusion-based algorithm.

A combination of subjective and objective evaluations was performed by Rao [3]. He used his synthesis model to implement a recognition scheme, in analysis by synthesis. He also demonstrated the distances between some original and the synthesized sample characters on a graph and reported the natural and legible appearance of the results. The results of character synthesis are reported to be similar to their corresponding natural characters. The shape vectors used in that work achieve 94% success rate as recognition models.

The performance of CAPTCHAs is evaluated by low OCR recognition rates while preserving reasonable human legibility. Hence, both OCR and subjective evaluation methods are needed to evaluate CAPTCHAs [18], [67].

2.1.4 Linking Applications, Specifications and Evaluation Methods

Applications may drive specifications related to the outputs of synthesis systems such as the level, data type, and writer-style imitation aspects. Table 3 suggests specifications of the outputs of synthesis systems for some common applications of

synthesized handwriting along with some suitable evaluation methods. The script aspect is not shown because it directly follows from the application script.

Table 3. Output specifications and evaluation methods for some common applications.

Application	Level	Online / Offline	Writer-Specific?	Suitable Evaluation Methods
Word Spotting	Word	Offline	Application dependent	Objective: Retrieval accuracy/sensitivity rates
CAPTCHAs	Character string	Offline	No	Subjective/Objective: Human legible text with deteriorated OCR rate
Character recognition improvement	Text	Both	Usually not	Objective: Recognition success ratio Objective: Analysis by synthesis
Forgery detection	Words or text lines	Mostly offline	Yes	Subjective: Handwriting style experts Objective: Writer identification results Objective: Resemblance with a reference model
Calligraphic & aesthetic styles	Words or text lines	Offline	Style specific	Subjective: Evaluation by experts and non-experts
Personalization	Words or text	Offline	Yes	Subjective: Evaluation by non-experts Objective: Writer identification results

2.2 Review on Shape-Simulation Approaches

Shape-simulation approaches for handwriting synthesis model the shapes of handwriting units rather than the movements that produce them. Hence, they are more practical when online data is not available, *i.e.* when data acquisition means are not restricted to PC-tablets.

There are generation and concatenation techniques for shape-simulation. Generation techniques synthesize new instances for a given writing unit while concatenation techniques connect smaller scripting units into larger ones. Figure 12 shows a classification of shape-simulation techniques under the generation and the concatenation approaches.

Generation techniques can be subdivided into: perturbation-based, fusion-based, and model-based techniques. Perturbation-based techniques generate new synthesized

text by altering geometric features such as the thickness and slant of one input sample. Fusion-based techniques take two-to-few input samples and fuse them into new outputs that take patterns from each input sample. Model-based techniques capture the variations in writing from many samples of a desired unit into models.

Concatenation techniques can be subdivided, according to the concatenation means they adopt, into no-connection, direct-connection, and modeled-connection. No-connection techniques juxtapose writing units into text lines. Direct-connection techniques take writing units and position them such that the ending ligature from one unit (also referred to as tail [44], [77], [78] or prefix segment [3]) directly connects to the starting ligature of the next unit (also referred to as head or suffix segment) to form a text line. Modeled-connection techniques add new connection ligatures synthesized by parametric curves.

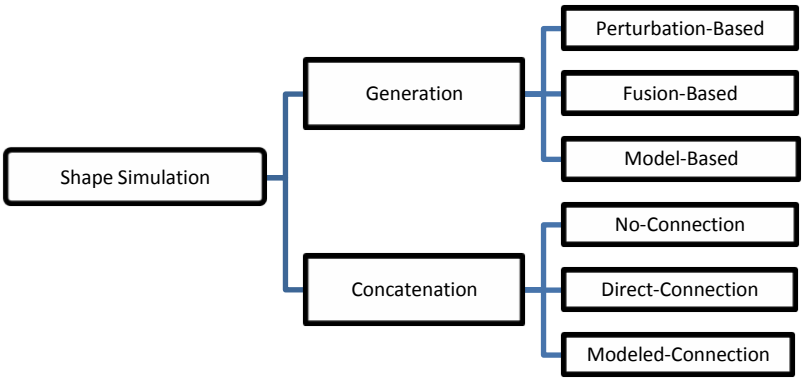


Figure 12. Classification of shape-simulation synthesis techniques.

Table 4 classifies common shape-simulation works, with the type of data and scripting units used. For character synthesis, generation techniques are more popular although concatenation was used to synthesize from characters from sub-characters [3]

[46]. On the other hand, cursive PAWs are mainly concatenated except when they are part of complete lines which are generated using perturbation [38]. For text line synthesis, both concatenation as well as generation techniques are commonly used although no work is reported on online synthesis of text lines using generation techniques. In Section 2.2 and Section 2.2.2, we discuss generation and concatenation techniques, respectively.

Table 4. Shape-simulation techniques with input/output units and scripting systems.

Output Unit / Data Type	Character		PAWs/Word		Text Lines	
	Concatenation	**Generation**	**Concatenation**	**Generation**	**Concatenation**	**Generation**
Online	Rao [3] Jawahar and Balasubramanian [46]	Stettiner and Chazan [65] Wang *et al.* [28], [44], [60] Choi *et al.* [45], [55] Zheng and Doermann [39]	Rao [3] Wang *et al.* [28] Saabni [14], [37]	---	Guyon [4] Lin *et al.*[13] Jawahar and Balasubramanian [46]	---
Offline	---	Mori *et al.* [5] Viswanath *et al.* [42], [71], [72] Miyao [12], [58] Thomas and Govindaraju [18], [67]	Helmers and Bunke [10] Thomas and Govindaraju [18], [67] Elarian *et al.* [56]	---	Elarian *et al.* [56]	Varga and Bunke [11], [38], [41] Vincent *et al.* [34] Cheng and Lopresti [73] Chen *et al.* [69]
Mixed	---	Dolinsky and Takagi [76][a] Liu *et al.* [50][b]	---	---	---	Xu [70] [c]

[a] Printed and online inputs, offline output

[b] Online input and offline-like output

[c] Not specified

2.2.1 Generation Techniques

As mentioned before, there are perturbation-based, fusion-based and model-based generation techniques. Perturbation-based techniques can disturb a single handwritten sample into several variations of it. Fusion-based techniques fuse two or more samples of

a unit shape into novel samples. Model-based techniques rely on large numbers of samples to generate models of a writing unit. Except for perturbation-based techniques, the two other techniques require shape-matching operations [39], [71]. Table 5 presents different works classified by the three generation-based techniques along with the various output data types used. In the following subsections, each of the three generation techniques is discussed in detail.

Table 5. Generation-based synthesis techniques with data types.

Technique / Data Type	Perturbation-based	Fusion-based	Model-based
Online	Lin and Wan [17] Wang *et al.* [44]	Zheng and Doermann [39]	Rao [3] Stettiner and Chazan [65] Choi *et al.*[55] Wang *et al.* [28], [44]
Offline	Varga and Bunke [11], [38], [41] Cheng [73] and Chen [69] with Lopresti	Viswanath *et al.*[42], [71], [72]	Mori *et al.* [5] Vincent *et al.* [34] Dolinsky and Takagi [19]
Mixed	---	---	Liu *et al.* [50][a] Miyao [12], [58][b]

[a] Online and offline
[b] Online inputs – offline outputs

Perturbation-Based Generation

Perturbation-based techniques generate new samples by altering geometric features such as the size, thickness and slant of a given sample. Perturbation-based operations can be seen as the inverse of the preprocessing steps employed in text recognition. Perturbation-based techniques are easy to apply, but the results may be unnatural due to random and non-calibrated parameter settings [11], [69], [73].

Stroke-wise rotation and scaling perturbations are applied to online strokes with high curvature points in [17]. Perturbations are added to text lines in [44] in order to generate additional training data to increase the variability within the dataset. Varga and

Bunke [11], [38], [41] apply non-linear geometric perturbations on complete text lines and connected components of offline images. They choose the parameters of their perturbation models randomly from predefined ranges. Their results show that this approach can be useful in improving hungry-for-data OCR recognition performance by adding synthesized data to otherwise small training sets. Cheng and Lopresti [73] calibrate the parameters of the perturbation-based model of the work of Varga and Bunke [38]. Chen *et al.* use those perturbation models for writer identification on Arabic handwritten data [69].

Fusion-Based Generation

Fusion-based techniques take few input samples and combine them into new synthesized outputs. They differ from concatenation techniques in that they generate scripting units at the same level as their inputs; *e.g.* characters generate new characters. Shape-matching algorithms are necessary for fusion-based techniques to make sure that segments are properly aligned. The number of unique outputs is limited in fusion-based techniques as compared to that of other generation techniques.

Zheng *et al.* [39] present a point-matching algorithm and apply it to generate online Latin characters by displacing the points in the range between two samples. Viswanath *et al.* [42], [71], [72] implicitly combine different partitions of samples of offline images into hybrid images while fixing their shared components. Fusion-based handwriting synthesis is not very common in the literature, probably because it is not as established as model-based techniques.

Model-Based Generation

Model-based techniques capture the statistics of natural handwriting variations into models. Although model-based techniques are profoundly established in theory, they may often be challenging to implement due to the large number of samples they require [39]. Models resulting from these techniques can also be utilized in recognition systems [3], [79]. Synthesis via model-based techniques can be seen as a decoding process after a lossy-compression encoding of many natural samples.

Model-based generation may process sampled points of data often chosen for their structural features *e.g.* maximum curvature [44] or zero-velocity [65], by spatial sampling *e.g.* equidistance [19] or by drawing them from a generative statistical recognizer *e.g.* a Bayesian network [55]. A common modeling scenario is that statistics on displacements of the sample points from a template sample are captured. New sample points are then drawn from the statistical model to generate shapes.

Techniques adopted for model-based generation depend, again, on the target applications and data types. In the following, we discuss the various techniques for model-based generation under online and offline categories.

Techniques that use online data

As for online data, different techniques are used to sample the drawn co-ordinates. One can extract straight graphemes within online characters and select them to be control points [3]. From these control points, more significant ones can be selected using Gabor filters [44] or Principle Component Analysis (PCA) [28]. Some works avoid the sampling of points and generate the co-ordinates directly [55], [65].

Once control points are selected from the online data, characters can be synthesized by using polynomial splines by connecting the control points [3]. One approach [44] is to match the control points to a template that is computed from all the sample characters and draw the control points according to a generative model of their displacements from the template and then using curves (splines) to connect them into a character-shape. Some authors have used Eigen vectors instead of splines [28].

Techniques that do not directly rely on the extraction of control points from sample characters, define generative models from which new samples can be synthesized. Some authors use generative statistical systems to synthesize handwriting through sampling from estimated joint distributions [55]. Stettiner and Chazan [65] consider the online x- and y-sequences of single-stroke character-shapes as the impulse response of an online signal. Characters are sampled into fixed sized vectors and match the points by using the Modified Newton Method. They find the character synthesizing filters by solving the optimization problems of the transfer functions for each pair of inputs and matched outputs.

Techniques that use offline data

These techniques work on the images of handwritten texts. A natural idea is to derive some template patterns from the offline data and then generate new samples from the templates. In [5], all the points from a sample of training data are matched with its class template and their displacements are recorded. Then generation of new samples is done by selecting new points within the pre-calculated displacements. A similar approach of generating samples from templates with displacements is used in [19]. However, the

authors used characters from standard fonts as templates. To calculate the displacements, the outlines of font templates are sampled equidistantly to match it with the offline images.

In another approach, Vincent *et al.* [34] applied fractal decomposition and synthesis as a lossy encoding-decoding process to offline character images. They defined reference bases that are repeated in an alphabet and then used these to model characters of the alphabet.

Techniques that use mixed online and offline data

There are several works that try to take benefits of both online and offline data. In [12], [58], affine-perturbed online data are thickened into offline data. All online samples of the Hiragana character set were optimally matched to a selected template sample by dynamic programming. The differences between the template and the other samples were modeled by PCA and the highest Eigen valued vectors were used for online sample synthesis.

Liu *et al.* [50] patented the idea of using trained Hidden Markov Models (HMMs) as generative statistical models to synthesize handwritten samples. The HMMs were trained as handwriting recognizers using handwritten and calligraphic-font samples. Pressure and ink data provided online and offline flavored outputs.

2.2.2 Concatenation Techniques

Concatenation refers to any synthesis approach that combines input samples into outputs of higher semantic levels. One common example is the concatenation of

character-shapes into words or text lines. Concatenation can be seen as the reverse of character segmentation in a text recognition system. It encompasses tasks such as baseline detection, horizontal space modeling, connection part segmentation and modeling, and segment joining and trimming. The input units for concatenation techniques are usually characters [13] but can also be sub-characters [3], character groups [4] or connected components [56].

Concatenation techniques depend on knowledge of the rules of a writing script. Some scripts, such as Arabic, enforce most characters to be joined in a continuous flow [80] while other scripts, such as the composite style of Latin, allow the writer to connect or disconnect characters. Others, such as Chinese, do not usually connect characters together.

The shape of the segments connecting characters, referred to as ligatures in [18], also relies on the script. In Latin, they often ascend in a curvy line to connect the suffix segment of a character to the prefix segment of the subsequent character [3]. The Arabic connection (Kashida) is usually horizontal with occasional vertical ligatures [56]. Concatenation techniques can be classified into no-connection, direct-connection, and modeled-connection. Table 6 shows works for online and offline no-connection, direct-connection, and modeled-connection categories.

Table 6. Classification of works according to concatenation techniques and data types

Concatenation Technique **Data Type**	**No-Connection**	**Direct-Connection**	**Modeled-Connection**
Online	Guyon [4][a] Jawahar and Balasubramanian [46][b]	Saabni [37]	Rao [3] Wang *et al.* [28], [44] Lin and Wan [17]
Offline	Elarian *et al.* [56]	Elarian *et al.* [56]	Thomas *et al.* [18]

[a] Sequential juxtaposition. Inking effect is added in one of the variations to generate an offline-like version of this approach
[b] Overlapping juxtaposition

No-Connection Concatenation

No-connection techniques concatenate scripting units by aligning them in juxtaposition without connection. Guyon [4] suggests simple juxtaposition of selected character strings to synthesize semi-cursive text. Character groups are selected based on their frequency in a linguistic corpus. In the training phase, a sample of each of the character strings is collected from the writer whose handwriting is to be imitated on an online tablet. In the synthesis phase, the text to be synthesized is parsed into a sequence of available character strings and the corresponding character string images are placed as text lines and paragraphs. This approach works well in subjective tests at the first glance. However, the trained eye may soon notice abrupt pen lifts between glyphs, repetitions of glyph appearance, and too regular pressure or inking. Geometric transformations are introduced to reduce such effects. In [56], non-connecting PAWs (Parts of Arabic Word) are aligned without any connection.

Direct-Connection Concatenation

Direct-connection techniques take writing units and position them such that the ending ligature from one unit directly connects to the starting ligature of the next unit to form text lines. These techniques are suitable for inherently cursive scripts like Arabic. Arabic online handwritten samples have been segmented and later concatenated to produce new samples in [37]. Similar ideas for segmenting, sampling and concatenating Latin characters were proposed in [17], [28], [44] and patented in [13]. In [56], samples of offline Arabic segmented characters are conditionally selected and later connected directly using the horizontal connection stroke (Kashida).

Modeled-Connection Concatenation

Modeled-connection techniques add new connection ligatures synthesized from models such as parametric curves. Rao [3] modeled the connection between the suffix segment of a character to the prefix segment of the subsequent character using polynomial and Bezier curves. His results of character to character concatenation are reported to appear natural, provided the segments of characters are adequately extracted.

Wang *et al.* [44] and Xu *et al.* [70] developed a character concatenation model in addition to a character generation model. Their character concatenation technique is similar to that of [3]: They concatenate the tail segment of a character to the head segment of the subsequent character (corresponding to the suffix and prefix segment in Rao's work, respectively) to minimize energy in a deformable model.

Style preserving concatenation [17] suggests connecting English characters according to some probabilities that reflect the writer's style. Whenever it is decided that characters should be connected, the extensions (probably trimmed) are connected with interpolation. If it is decided that characters should not be connected, an ending-position, rather than a middle-position, sample of the character is used (*i.e.* a no-connection technique).

Cursive handwritten CAPTCHAs are produced by the concatenation of skeletonized characters at the level of the baseline [67]. They define their connection ligatures by looking at the derivative of the vertical projection. They parameterize ligatures and join them from the end of a character to the body of the next character. Table 7 summarizes some key shape-simulation works.

Table 7. Summary of the specifications of shape-simulation systems.

Author(s) [citations]	Technique	Input unit	Output unit	Online/ offline	Applications	Evaluation	Script
Rao [3]	Concatenation	Characters	Cursive writing	Online	OCR Data compression	Subjective and analysis by synthesis	Latin
	Model-based	Sub-characters	Characters				
Stettiner and Chazan [65]	Model-Based	Character	Character	Online	OCR, Writer-imitation & identification	Test of completeness for the model	Latin
Guyon [4]	Concatenation	Glyphs	Semi-cursive writing	Online + inking	Personalization, Pleasant view	Subjective	Latin
Mori *et al.* [5]	Model-based	Digits	Digits	Offline	OCR	---	Digits
Wang *et al.* [44]	Concatenation	Ligatures	Cursive writing	Online Online	Pen-based computers	Subjective	Latin
	Model + some Perturbation	Characters	Characters				
Wang *et al.* [28]	Concatenation + sampling	Characters with extensions	Cursive writing	Online	Pen-based computers	---	Latin
	Model-based	Segmented samples	Characters				
Helmers and Bunke [10]	Concatenation	1.Isolated characters 2.characters form text 3.*n*-tuples of characters	Cursive writing	Offline	OCR	OCR	Latin
Choi *et al.* [45], [55]	Model-based	Characters	Characters	Online	Personalization	Subjective	Hangul & Digits
Varga and Bunke[11], [38], [41]	Perturbation	1) text line 2) connected components	1) Text line 2) connected components	Offline	Training data for HMM-based OCR	OCR training	Latin
Viswanath *et al.* [42], [71], [72]	Fusion-based	Characters	Characters	Offline	Nearest Neighbor Classifier	Nearest Neighbor Classifier	Digits
Zheng and Doermann [39]	Fusion-based	Characters	Characters	Online	OCR, Writer Identification	Deformation error	Latin
Vincent *et al.* [34]	Model-based	Text line	Text line	Offline	Scripting system recognition, Compression	Language and Script Recognition	Latin Arabic Cyrillic
Miyao [12], [58]	Model-based	Online characters	New offline characters	Online	Offline OCR	OCR training	Hiragana
Dolinsky and Takagi [19], [66], [76]	Model-based	* font character *handwritten samples	Characters	Both	Human-like behavior, personalized PC	Errors of recurrent neural networks	Hiragana
Lin and Wan [17]	Concatenation + sampling	Characters	Cursive writing	Online	Aesthetical & personal view, forensics, For disabled, OCR, CAPTCHA	Subjective	Latin
	Perturbation	Characters	Characters				
Thomas and Govindaraju [18], [67]	Concatenation	Characters	Cursive writing	Offline	CAPTCHA	Subjects and OCR performances	Latin
	Perturbation	Characters	Characters				
Elarian [56]	Concatenation + sampling	Segmented characters	Text lines	Offline	OCR, Word-Spotting	Comparison between *best* and *worst* synthesis	Arabic

Saabni and El-Sanaa [14], [37]	Concatenation	Segmented characters	Connected components	Online	Holistic OCR	OCR training	Arabic
Cheng and Lopresti [73]	Perturbation	Text line	Text line	Offline	OCR	Subjective	Latin
Chen *et al.* [69]	Perturbation	Text line	Text line	Offline	Writer identification	Writer Identification	Arabic
Jawahar and Balasubramanian [46]	Model + Concatenation	Characters	Characters and words	Online	OCR, personalization, study of human style	Subjective	Indian (Hindi, Tamil, Malayalam, Telugu)
Fujioka [59]	Generation & concatenation	Character	Cursive writing	Offline	Calligraphy	Subjective	Japanese (Kana)
Xu [70]	Generation & concatenation	Text	Novel cursive text	Unspecified	Personalization	NA (Patent)	Latin
Liu *et al.* [50]	Model	One or more characters	Characters probably with inking	Online→ offline-like	Personalization & artistic view	NA (Patent)	Latin
Lin *et al.* [13]	Concatenation	Character positions	Text line	Online	Training OCR	NA (Patent)	Latin

2.3 Overview on Some Other Synthesis Approaches

In this section we present techniques for handwriting synthesis which are non-shape simulation approaches. The most common of the non-shape simulation approaches are the group of techniques which can be termed movement-simulation approaches. Movement simulation is a top-down approach to handwriting synthesis where the neuromuscular acts of writing are simulated. One approach to synthesizing handwritten data is to model strokes as oscillatory components where the character formation is a result of horizontal and vertical oscillations (*i.e.* constrained modulation); the horizontal oscillation and its modulation controls the stroke/character-shape and the vertical oscillation and its modulation controls the character height [78]. Motivated by this idea, Gangadhar *et al.* proposed a neural network model of handwriting strokes, where the stroke velocities are expressed as oscillatory neural activities. The architecture has stroke selection as the input layer and the estimated stroke velocities are represented by the output layer [30].

One of the most notable contributions for modeling strokes is by Plamondon and his group [31]–[33], [64], [81]–[83]. The strokes are defined from the context of Kinematic Theory of Rapid Human Movement as primitive movement units which can be superimposed to construct word patterns [40]. A stroke model describes the essential characteristics of the pen-tip trajectory [84]. The main idea behind the Kinematic Theory is that a neuromuscular system involved in the production of a rapid movement can be considered as a linear system made up of a large number of coupled subsystems and the impulse response of such system converges toward a lognormal function under certain conditions [82], [83], [85].

There are many models derived from this lognormal paradigm. These models can be broadly categorized into two:

(i) Delta-Lognormal, which involves two neuromuscular systems (each described by a lognormal impulse response and timing properties), one agonist to, and the other antagonist to, the direction of the movement. This model generates straight strokes and predicts all the velocity patterns observable in a set of strokes.

(ii) Sigma–Lognormal model, where the assumption is that the two neuromuscular systems do not work in exactly opposite directions and thus the resultant velocity is described by the vectorial summation of the contribution of each of the neuromuscular systems involved. Further in sigma-lognormal models, there are two versions: a straight vector (the simpler version) and a curved vector (a more complex but precise version where it is assumed that the input command vectors are not straight but

curved). The curved sigma-lognormal models can be used to generate single strokes with almost any required precision, depending on the number of parameters used.

All the different models differ in their stroke generation quality depending on the number of parameters used in a given model (the simple one with three parameters to the more complex ones having up to 11 parameters) [64].

Estimating the parameters robustly is one of the issues in using these stroke models for handwriting synthesis. Moreover, the variability of handwriting, as a result of varying the parameter values, to generate realistic text needs further investigation. There are many methods proposed to estimate the initial parameters of the log-normal stroke models [84], [86]. The INFLEX algorithm exploits the characteristics of the tangent lines at the inflexion points of a single lognormal to estimate the initial parameter values. Later, it uses non-linear regression to optimize the initial solution (minimizing mean square error). The INITRI algorithm [87] uses analytical methods to estimate the initial parameters. Two points are selected along the rising velocity curve (it is assumed that mainly the agonist component contribute during the increasing part of the velocity curve) along with the time occurrence of the maximum velocity and the relationships between the parameters to estimate the initial values. This is later optimized using non-linear regression. Further, a third algorithm named XZERO is proposed that exploits the analytical relationships existing between three points of the lognormal profile *i.e.* maximum (the first order time derivative is zero) and two inflexion points (the second order time derivatives are zero). Each of the above three algorithms has its advantages

and limitations, and the authors have proposed using hybrid versions of them as they seem complementary to each other [87].

In [40], the authors presented a system for synthesizing a large database of handwriting from few specimens using the Sigma-Lognormal model. The system can be used to generate writing samples for a single writer, as well as in multi-writer setup. The variability observed in handwriting data can be regenerated by varying the Sigma-Lognormal parameters around their mean values within the limits fixed by their standard deviations. The factor of variability needs to be carefully fixed so as to get intelligible samples.

In another approach, time trajectories of the English alphabet were modeled using oversampled reverse time delay neural network (TDNN) architecture to generate outputs that can control the writing of characters with a pen [88]. The network was trained on character glyphs as a sequence of successive points in time. Three outputs provided the time sequences of signals that controlled the X and Y positions of the pen and up/down pen control.

Bayoudh *et al.* [35] propose using the principle of analogical proportion to synthesize new examples from an existing limited set of real examples. Each character is represented as a sequence of Freeman chain codes including a set of anchorage points. Experiments evaluated the improvement in the training of a set of classifiers on character recognition rate as a result of increasing the size of the dataset. The results confirmed that the proposed approach is as effective as character synthesis through knowledge-based

approaches in the form of image-based (scant and slat) distortions and online (speed and curvature) distortions.

Slim and Benrejeb [89] modeled the handwriting process of few Arabic characters using electro-myographic signals (EMG) generated by muscles in the forearm. An RBF neural network with feedback and time delay learns to associate the EMG signals generated, as a character is drawn, with the sequence of pen displacements recorded in the X and Y directions. Inverse models are also described for generating the EMG signals from the recorded position signals.

2.4 Synthesis for Text Recognition

Synthesis based on the kinematic theory and on shape-simulation can be used to improve text recognition in terms of recognition accuracy [10], [35], stability with new classes [90], [91], and speed performance [58], [92]. Bayoudh *et al.* [35], for example, expanded the training set of a recognition system and achieved improvements on the character recognition rate for their online test set.

Shape-simulation via perturbation-based, fusion-based and model-based generation [39] were also used to enhance recognition accuracy. Varga and Bunke [11], [38], [41], for example, apply geometric perturbations on handwritten text-lines to supplement training sets of recognition systems. Similarly, Wakahara *et al.* [93] Keysers *et al.* [94] apply affine transformations and local perturbations for the same goal, respectively. Fusion-based techniques combine two samples into shapes that take features from both inputs. Zheng and Doermann [39] and Viswanath *et al.* [42], [71], [72] adopt

fusion-based techniques for the expansion of training sets. Model-based techniques are used for online recognition in [28], [55], [65] and for offline recognition in [5], [19].

Concatenation operations can be performed, with or without connecting the aligned units, for the same goal. It was used without to form words and lines for a training set in [4], [46] and [56]. Direct-connection techniques connect character tails to their heads, as in [37] [56] for Arabic and [17], [28], [44] [13] for Latin cursive text-lines. More sophisticated concatenation was achieved by connection-stroke interpolation which is based on polynomial-models [46], modeled-models [44][3][70] or probabilistic-models [10][17].

Bayoudh *et al.* [35] inject 300 synthesized versions of the 26 English characters to the training set and increase the character recognition rate (CRR) by up to 13%. Helmers and Bunke [10], Saabni and Sanaa [14], [37], and Miyao and Maruyama [12] generate data that performs approximately as well as their collected data for the recognition of Latin, Arabic and Hiragana, respectively. Varga and Bunke [11], [41] improve recognition rates of Latin handwriting by around 16% by injecting perturbed data. Similarly, Plamondon *et al.* [92] inject synthesized samples to reduce the error rates of a set of 11 online gestures by 50%.

2.5 Arabic Handwriting Synthesis

Movement-simulation for cursive handwriting, including Arabic words, is performed by superimposing velocity beta profiles of basic writing strokes [95]. Ltaief *et al.* propose neural networks to model curvilinear velocity beta profiles for Arabic and Latin [96].

As for shape-simulation, offline Arabic synthesis was first presented by Elarian *et al.* [56] where the idea of sample selection and concatenation was introduced. Online concatenation, after PCA reduction of the samples space, is addressed by Saabni and El-Sanaa [14], [37]. Dinges *et al.* [97], [98] generate and concatenate offline Arabic character-shapes from online data. Chen *et al.* use perturbation models for writer identification from Arabic handwriting [69].

We conclude from this chapter that for Arabic recognition enhancement, concatenation-based synthesis may have advantage over generation-based synthesis; since it can provide arbitrary vocabulary. Besides, when offline data is concerned, shape-simulation becomes handier than movement-simulation. Arabic concatenation requires no-connection techniques between PAWs and direct-connection or modeled-connection within them. We depict such conclusions in the highlighted cell of Table 8.

Table 8. The adequate specifications of synthesis systems per technique.

Technique	Movement-Simulation	Shape-Simulation
Generation	Closed Vocabulary, Online Data	Closed Vocabulary, Offline Data
Concatenation	Open Vocabulary, Online Data	Open Vocabulary, Offline Data

CHAPTER 3

ARABIC HANDWRITING ANALYSIS AND DATASET DESIGN

Handwriting synthesis necessitates the acquisition of samples that *cover* a writing system. *Coverage*, here, refers to the presence of sufficient samples to be capable to generate any arbitrary text in a given scripting system. Moreover, the samples may need preprocessing and preparation to enhance their usage. In this chapter, we analyze Arabic typographic models and ligatures. Then, we address the design and collection of a covering dataset for Arabic script.

3.1 Analysis of Arabic Typographic Models

The traditional Arabic typographic model contains a large number of character-shapes that may combine to create hundreds of ligatures. In order to reduce these numbers, we can use other models to merge resembling character-shapes into *groups* [99]. For example, the *dot-less* model [100], [101] divides Arabic character-shapes into groups that share identical character bodies with different stress marks (dots ".", Hamza "ء" and Madda "~"). Figure 13 shows a dot-less character group and some characters that participate in the group for the (B) and (M) character-shapes only.

Isolated Shape (A)			Ending Shape (E)			Middle Shape (M)	Beginning Shape (B)
ن	ي ئ	ب ت ث	ـن	ـي ـئ	ـب ـت ـث	بـتـثـيـئـنـ	بـتـثـيـئـنـ

Figure 13.Sample characters that differ only in dots/Hamza in some character groups.

The *2-Shapes* model [102], [103] represents the (B) and (M) shapes of a character by the (B) character-shape for most characters. It does so as the (M) shape resembles the (B) shape of the same character, except for an additional small extension to its right. Similarly, it represents the (A) and (E) shapes of most characters by the (A) character-shape for the same reason. The only exception for such resemblances occurs with the *Ain* and *Heh* character groups.

The *1-Shape* model benefits further from the resemblance between the (B)-shapes (M)-shapes from one side and the (A)-shapes and the (E)-shapes from the other side and the characters, except for some tail parts. In the literature [104], "root shape" is defined as the part of the character that is independent from its position. The "tail shape" is a curved extension that follows some root shapes (*i.e.* (A), (E)) at word-ends. If the tail shape is removed from the root shape, many characters can be represented with the single root shape. Table 9 shows one example of a character that only fits in the 4-Shapes model, a character that fits in 2-Shapes model, and a character that fits in the 1-Shape model.

Table 9. Examples of character-shapes with the name of the smallest model that applies to the set.

Model	Beginning		Middle		Ending		Isolated	
4-character-shapes		هـ		ـهـ		ـه		ه
2-character-shapes		كـ		ـكـ		ـك		ك
1-character-shape		جـ		ـجـ		ـج		ج

Figure 14 visualizes the reductions in the different glyphs models. The hashed bars show the number of shapes if the dot-less models is applied in addition to a shape-model.

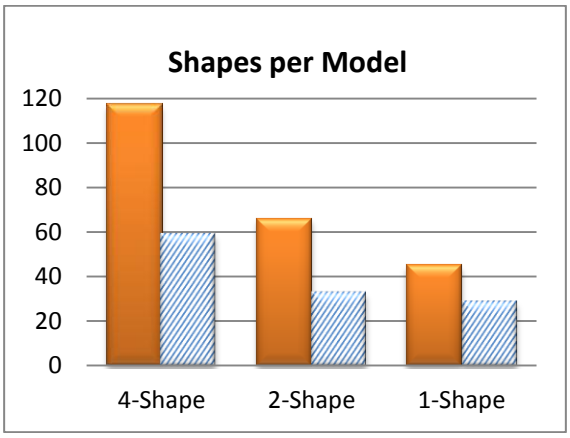


Figure 14. The numbers of shapes in different models of the Arabic writing. The hashed bars are for the dot-less versions of a model.

The dot-less model exploits resemblances among characters; whereas *the 2-Shapes* model exploits resemblances among shapes of characters. These two reduction-models are orthogonal; *i.e.* they can be combined. The three reduced typographic models are shown in Table 10.

Table 10. Arabic characters grouped based on the dot-less, the 2-Shapes, and the combined models.

Dot-less Model

A	E	M	B
ء			
ا أ إ آ	ا أ إ آ		
ب ت ث	ب ت ث	ب ت ث ن ي ئ ى	ب ت ث ن ي ئ
ن	ن		
ي ئ ى	ي ئ ى		
ج ح خ	ج ح خ	ج ح خ	ج ح خ
د ذ	د ذ		
ر ز	ر ز		
س ش	س ش	س ش	س ش
ص ض	ص ض	ص ض	ص ض
ط ظ	ط ظ	ط ظ	ط ظ
ع غ	ع غ	ع غ	ع غ
ف ق	ف ق	ف ق	ف ق
ك	ك	ك	ك
ل	ل	ل	ل
م	م	م	م
ه ة	ه ة	ه	ه
و ؤ	و ؤ		

Underlined character-shapes represent their groups, when more than one character is in a cell.

2-Shapes Model

A	E	M	B
ء			
ا أ إ آ			
ب ت ث ن ي ئ ى		ب ت ث ن ي ئ ى	
ج ح خ		ج ح خ	
د ذ			
ر ز			
س ش		س ش	
ص ض		ص ض	
ط ظ		ط ظ	
ع غ	ع غ	ع غ	ع غ
ف		ف	
ق		ق	
ك		ك	
ل		ل	
م		م	
ه ة	ه ة	ه	ه
و			
ؤ			
ي		ي	
ى			
ئ		ئ	

Combined Model

A	E	M	B
ء			
ا أ إ آ			
ب ت ث ن ي ئ ى		ب ت ث ن ي ئ	
ج ح خ		ج ح خ	
د ذ			
ر ز			
س ش		س ش	
ص ض		ص ض	
ط ظ		ط ظ	
ع غ	ع غ	ع غ	ع غ
ف		ف	
ق		ق	
ك		ك	
ل		ل	
م		م	
ه ة	ه ة	ه	ه
و ؤ			

The counts of character-shapes for the traditional and reduction-models are displayed in Table 11. These counts are later considered in the design of our ligative and unligative forms.

Table 11. Numbers of character-shapes for different typographic models.

Model	Isolated Shape (A)	Ending Shape (E)	Middle Shape (M)	Beginning Shape (B)	Total
Traditional	36	35	23	23	117
Dot-Less	19	18	11	11	59
2-Shapes	35 + 5[a]		23 + 3[b]		66
Combined	18 + 3[c]		11 + 2[d]		34

[a] Corresponding to the extra (A) shapes of *Hamza*, *Ain*, *Ghain*, *Heh* and *Teh Marbuta*.
[b] Corresponding to the extra (M) shapes of *Ain*, *Ghain* and *Heh.*
[c] Corresponding to the extra (A) shapes of *Hamza*, *Ain* and *Heh*.
[d] Corresponding to the extra (M) shapes of *Ain* and *Heh*.

The use of reduced typographic models is especially handy when designing ligative datasets. This is because the ligative dataset covers bigram combinations of character-shapes, the number of which (2,622, as in Appendix D) is of quadratic order of the underlying alphabet whereas the unligative dataset covers single character-shapes.

3.2 Analysis and Design of Dataset

Part of this work is to design an Arabic handwritten dataset to serve the objectives of this dissertation, as well as other research objectives. In the discussion below, we analyze and elaborate on the design of an Arabic handwriting dataset suitable for synthesis.

We design a dataset that consists of *parts* each of which aims at ensuring some kind of coverage. The covering units of the different parts of the dataset range from isolated characters to paragraphs and contain units like isolated bigrams, words and sentences. In general, the design of all dataset parts emphasizes on their conciseness and adequate level of naturalness. Hereafter, we will use the acronym(s) PoD(s) to abbreviate "Part(s) of the Dataset".

In the following sections, we introduce a systematically designed set of separate ligative and unligative texts used for the collection of handwriting samples. In addition, we briefly present two other dataset parts that were aggregately collected.

3.2.1 The Ligatures Part of Dataset

Using ligatures may significantly change the shape of one or more characters. Hence, ligature identification and distinction is useful. Comprehensive datasets of aligned text and images, which are necessary for the development of automatic text recognition and handwriting synthesis systems [56], [105], [106], have started including ligature information in their ground-truths. Some modern Arabic datasets, *e.g.* ERIM [107] and IFN/ENIT [108], recognize the importance of ligature identification in ground-truths by assigning some of the common ligatures distinct encodings. However, ligature identification necessitates laborious human intervention [104]. We separate ligative from unligative texts to ease ligature identification in our datasets.

Arabic script calligraphic workbooks [109], [110] suffer from the absence of an explicit and comprehensive list of ligatives. Such a list is useful for font development, dataset design, text recognition, and text synthesis research. Researchers have reported encountering more than 200 distinct bigram and trigram Arabic ligatures [111], which is a sizable number [112]. However, these ligatures are not systematically documented. The Unicode standard [113] contains more than 300 ligatures. However, it often lacks consistency as the Unicode standard frequently defines a ligature for a pair of character-shapes while ignoring similar cases for character-shapes that may only differ from the defined pair by dots (*i.e.* they share a dot-less model).

Optional ligatures may occur when characters connect into a shape that differs from the horizontal Kashida concatenation of their shapes. The ligatures part of the dataset (PoD) is dedicated to gather isolated bigrams and words that can optionally contain ligatures. Ligatures are *n*-grams in essence; hence, the number of their possible combinations grows exponentially with the number of their composers.

A ligature may only occur if a character connects to its subsequent. Hence, bigram ligatures can be formed by either a (B) or an (M) character-shape followed by either an (M) or an (E) character-shape. In regular expressions, these are denoted as: <(B)(M)>, <(B)(E)>, <(M)(M)>, and <(M)(E)>. These bigrams can be considered as (B)-ligature shapes, (A)-ligature shapes, (M)-ligature shapes and (E)-ligature shapes, respectively.

Comprehensiveness of the Ligative Part

We aim at making a comprehensive list of bigram ligatives and then develop a rule that extends it to *n*-gram ligatives. Bigrams occur when a (B) or an (M) shape is followed by an (M) or an (E) shape. Table 12 shows Arabic bigrams. Each row corresponds to a (B) or an (M) character group, identified by a representative character, according to the dot-less model. Similarly, each column corresponds to an (M) or an (E) character group. This generates four expressions for bigrams: <(B)(E)>, <(B)(M)>, <(M)(M)>, and <(M)(E)>, each of which is located in a quadrant in the table. The numbers shown in Table 12 are the counts of ligatives according to the traditional model. These are computed as the products of the group cardinalities of the row and columns to which they belong.

Table 12. Bigrams based on the dot-less model with shadings on the shortest 1-shape representatives.

Ending (E)																	Middle (M)											Dot-less Groups	
ي	و	ه	ن	م	ل	ك	ق	ف	ع	ط	ص	س	ر	د	ج	ب	ا	هـ	مـ	لـ	كـ	فـ	عـ	طـ	صـ	سـ	حـ	بـ		
18				6											18			6	6								18		بـ	Beginning (B)
				3											9			3	3								9		حـ	
6				2									4		6			2	2								6		سـ	
6				2									4		6			2	2								6		صـ	
				2											6			2	2								6		طـ	
				2											6			2	2								6		عـ	
6				2											6			2	2								6		فـ	
3				1	1										3		4	1	1	1							3		كـ	
3				1											3		4	1	1								3		لـ	
				1											3			1	1								3		مـ	
				1											3			1	1								3		هـ	

18			6	6									12		18			6	6								18		بـ	Middle (M)
				3											9			3	3								9		حـ	
6				2									4		6			2	2								6		سـ	
6				2									4		6			2	2								6		صـ	
				2											6			2	2								6		طـ	
				2											6			2	2								6		عـ	
				2											6			2	2								6		فـ	
				1	1										3		4	1	1	1							3		كـ	
				1											3		4	1	1								3		لـ	
				1											3		4	1	1								3		مـ	
				1									2		3		4	1	1								3		هـ	

In general, *n*-gram ligatives require a PAW to contain $n - 1$ overlapping ligatives. Overlapping ligatives refer to consecutive ligatives that share a connecting (M) character. For example, the word "محمد" has a trigram ligative "محمـ" that is formed by combining the bigram "محـ" with the bigram "حمـ" using Table 12, with character "حـ" being the connecting character.

Compactness of the Ligative Part

When designing a dataset, one often looks for a compact, yet comprehensive dataset. The compactness of an Arabic dataset can be achieved by reducing the numbers of character-shapes and PAWs; since PAWs are the smallest scripting units that bear information on character connections [16], [114], [115]. We define the *character*- and *PAW-bounds* (abbreviated as CB and PB) as the minimum numbers of character-shapes and PAWs required by a comprehensive dataset, respectively.

Ligatives taken from the topmost left quadrant of Table 12, viz. <(B)(E)>, are *standalone-ligatives* as they are written without being connected to previous or subsequent characters. It is more natural to write *standalone* bigrams in isolation than it is for other bigrams. Therefore, we use them to represent all bigrams in the other quadrants, which corresponds to the 2-Shapes reduction-model. The ligatives that are highlighted in Table 12 are those that do not have standalone representatives. Hence, we insert them into words, as shown in Table 13 to be naturally collected in our dataset. Again, we use <(B)(M)> bigrams to represent corresponding <(M)(M)> bigrams, in conformance with the 2-Shapes model.

Table 13. Example words containing ligatives that do not have standalone bigrams.

<(M)(E)>			<(B)(M)>													
بر	بن	ـهر	ـهأ	ـما	ـهه	ـمه	ـلهـ	ـكه	فهـ	عهـ	طهـ	صهـ	سهـ	حهـ	بهـ
جِبْر	لَبَن	نهر	مها	لما	ههنا	مها	لَهْو	گَهْل	فهم	عهن	طهر	صَهْوَةُ	سُها	جهل	بِها
ستر	فِتْنٌ	بهز							قَهْر	بلاغها	ظهر	أرضها	شهي	كفاحها	تَهَبُ
نثر	بَحَثْنَ													بذخها	نفاثها
كَنَر	مَنَن														نَهار
خَيْر	تين														يَهَبُ
بِئْر	يَنَنَّ														فنائها
خُبز															
يَتَزَگَى															
گَنْز															
ضِيزَى															
يَشْمَئِزّ															

In Table 14, we display PB and CB under the four typographic models. PB is computed from Table 12 as follows: In the traditional model, we sum up all the numbers in the table; in the dot-less model, we count the number of filled cells; in the 2-Shapes model, we sum up the numbers in the <(B)(E)> quadrant, and expanded in Table 15, to those that are highlighted in the other quadrants; finally, in the combined model, we count the number of highlighted cells, and the filled cells in the <(B)(E)> quadrant.

Table 14. Bounds of ligatives for the four typographic models.

Model	<(B)(E)>		<(B)(M)>		<(M)(M)>		<(M)(E)>		Total	
Bound	**PB**	**CB**	**PB**	**CB**	**PB**	**CB**	**PB**	**CB**	**PB**	**CB**
Traditional	151	302	116	≥348	116	≥348	165	≥495	548	≥1493
Dot-Less	34	68	34	≥102	34	≥102	34	≥102	136	≥374
2-Shapes	151	302	23	≥69	23	≥69	28	≥84	225	≥524
Combined	34	68	11	≥33	11	≥33	5	≥15	61	≥149

Character-bounds in Table 14 can be found from the corresponding PB by the following relations: The character-bounds of <(B)(E)> bigrams are twice as much as their PAW-bounds. The character-bounds of the <(B)(M)> and <(M)(E)> ligatives are at least three times as much as their PAW-bounds. The character-bounds of <(M)(M)> bigrams are at least four times as much as their PAW-bounds.

Table 15. Isolated-ligatures expanded list.

33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
مم	مج	مم	مج	لي	لم	لج	لا	كي	كم	كل	كج	كا	ني	فم	فج	عم	عج	طم	طج	صي	صم	صر	صج	سي	سم	سر	سج	جم	جج	بي	بم	بج	1
	مح		مح	لئ		لح	لإ	كئ			كح	كإ	قي	قم	قح	غم	غج	ظم	ظج	ضي	ضم	ضر	ضج	شي	شم	شر	شح	حم	حج	تي	تم	تج	2
	مخ		مخ	ل		لخ	لأ	كى			كخ	كأ	فئ		فخ		عج		طج	صئ		صز	صج	سئ		سز	سخ	خم	خج	ثي	ثم	ثج	3
							لآ					كآ	قئ		قح		غج		ظج	ضئ		ضز	ضج	شئ		شز	شح		جج	بي	نم	نج	4
													ق		فخ		عخ		طخ	صى			صخ	سى			سخ		حج	بي	بم	بج	5
													قى		قخ		غخ		ظخ	ضى			ضخ	شى			شخ		خج	ئي	ئم	ئج	6
																													جخ	ئى		بج	7
																													حخ	تئ		تج	8
																													خخ	ثئ		ثج	9
																														نئ		نج	10
																														بئ		بج	11
																														ئئ		ئج	12
																														بى		بخ	13
																														نى		نخ	14
																														ثى		ثخ	15
																														ى		نخ	16
																														بى		بخ	17
																														ئى		ئخ	18
151	1	3	1	3	3	1	3	4	3	1	1	3	4	6	2	6	2	6	2	6	6	2	4	6	6	2	4	6	3	9	18	6	18	Total

3.2.2 The Unligative Text and the Isolated Characters Parts of Dataset

A comprehensive unligative dataset covers all character-shapes while avoiding ligatives. Pangrams, in logology [116], are texts that contain every character of an alphabet [117]. Lipogram are writings constrained to avoid sets of characters [118]. Hence, a comprehensive unligative dataset is essentially a special pangram with a special lipogram condition.

The unligative text (UT) PoD and the isolated characters (IL) PoD, together, cover all Arabic character-shapes and some obligatory ligatures. The idea of making minimal but meaningful texts that cover all possibilities of an Arabic writing unit is borrowed, in concept, from [106], where they select single words that cover all character-shapes. Unfortunately, some of the words they provide are awkward to ordinary writers. Besides, sentences and short stories can bear more features of the natural writing than single words (*e.g.* how writing inclines at different positions of a page). For these reasons, the UT and the IL parts were designed.

To ease keeping track of the numbers of the Arabic character-shapes in a text, a GUI tool, with a snapshot in Figure 15, was implemented. It distinguishes with colors the character-shapes that are used zero, one, or more times. The tool has a batch processor as well as a design window that displays statistics of the text that is edited. It also includes an option to exclude character-shapes that occur in a form that is ligaturisable from character-shape counts.

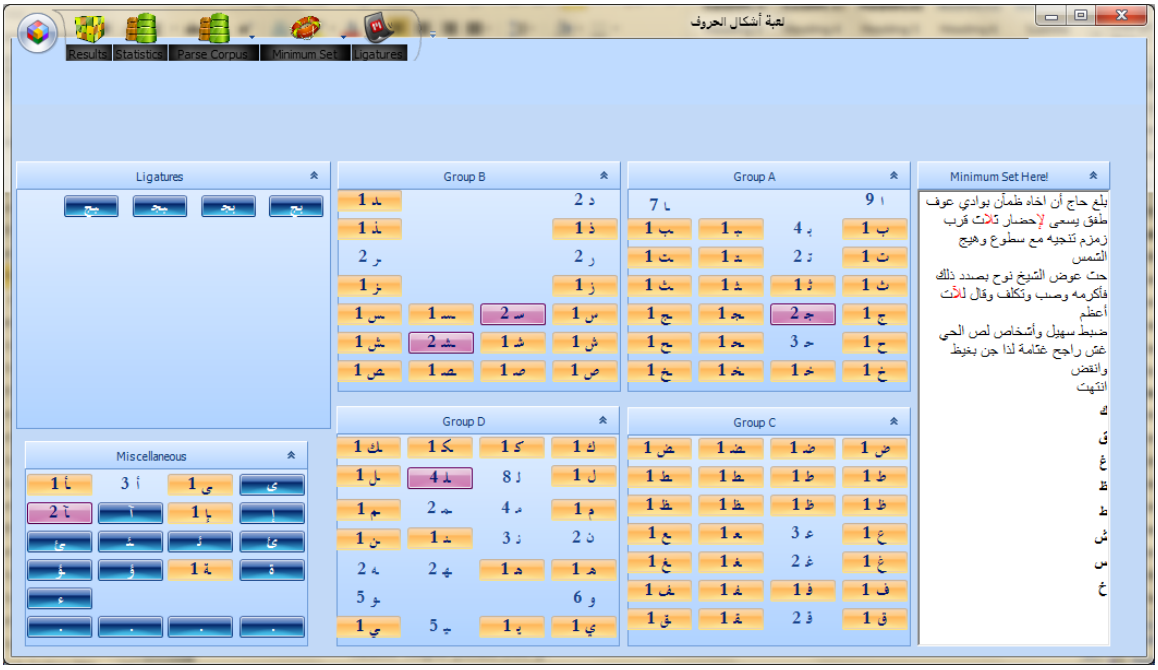


Figure 15. A snapshot of the GUI tool that counts character-shapes analyzing a text

Several character-shape pangrams were suggested for unligative forms (*e.g.* the one in Section 6.2). The set of sentences in Figure 16(a), along with the set of (A) character-shapes in Figure 16(b), were finally chosen to appear in all unligative forms. The separation of the eight character-shapes of Figure 16(b) helps reducing the total number of words in the dataset since these shapes can exist only once per word, at most. The pangram contains 43 words with 163 shapes that are shown in Figure 16.

بلغ حاج أن اخاه ظمآن بوادي عوف طفق يسعى لإحضار ثلاث قرب زمزم تنجيه مع سطوع وهيج الشمس
حث عوض الشيخ نوح بصدد ذلك فأكرمه وصب وتكلف وقال للآت أعظم
ضبط سهيل وأشخاص لص الحي
غش راجح غثامة لذا جن بغيظ وانقض انتهت

(a)

ك	ق	غ	ظ	ط	ش	س	خ		

(b)

Figure 16. (a) the selected Arabic character-shape pangram with obligatory (highlighted) and optional (underlined) ligatives, and (b) the complementary set of (A) character-shapes.

We choose not to tackle omni-ligatives in the forms and to develop semi-automatic tools that are particularly aware of their positions, instead.

Comprehensiveness of the Unligative Text and the Isolated Letters Parts

Although several Arabic character pangrams (known in Arabic as "جامع الحروف" [119]) are in the literature [120], we need a special kind of written pangrams that accommodate the occurrence of every character-shape in the writing. Our pangram condition can be asserted by ensuring that every instance of the 4-shapes model is included in the dataset.

We also need the pangram to conform to a special lipogram condition: to avoid the ligative *bigrams* of character-shapes. Our lipogram condition is assured by avoiding the usage of the ligative bigrams of Table 12. Unfortunately, the two conditions sometime cannot be fulfilled together because of what we call *omni-ligativ*es. Omni-ligatives are character-shapes that have the potential to ligate with every previous character. An omni-ligative is evident when a column of a character-shape is fully-populated. From Table 12, we can spot fully-populated columns corresponding to five omni-ligative dot-less character-shape groups (viz., ـح, ـحـ, ـم, ـمـ, and ـهـ).

We formulate the pangram selection problem as a Set Covering Problem [121], [122] and follow a greedy approach to find a (probably suboptimal) solution to it. Our Character-Shapes Covering algorithm (CSC), listed in Figure 17, selects a pangram for a given alphabet from a parsed corpus. A lipogram option can be set to ignore character-shapes that appear in ligative bigrams but are not omni-ligatives.

Moreover, a heuristic is used to help making such pangram compact. The heuristic favors the early coverage of character-shapes with few occurrences in the corpus. Iteratively, CSC computes a cost function for each input sentence based on the occurrence of the least frequent character-shape in it. The cost function considers the uncovered character-shapes that a sentence can add. The sentence with the minimum cost and fewest characters is added to the pangram and its character-shapes are overlooked in subsequent iterations. Eventually, if the corpus contains all character-shapes, the algorithm halts with a pangram.

Character-Shapes Covering (CSC)

Inputs:
R, a corpus of Arabic sentences
A, the alphabet of character-shapes
Include_lipo_condition, a flag to request lipogram monitoring

Output: ***P***, an Arabic pangram

Algorithm:

```
1. Initiate set S to contain all elements in A;
2. Let H be the histogram of the character-shapes in S based on
   probabilities computed from R;
3. Repeat until set S is empty
4.    For each sentence X of R
5.         Let x be the set of character-shapes in X;
6.                if Include_lipo_condition
7.             remove from x all character-shapes that only appear in
   ligative bigrams;
8.     Added_Value = x ∩ S;
9.     C_x = minimum(H{Added_Value});
10.           Move X with minimum C_x to the output set P resolving ties
   with the smallest X;
11.       Remove entries in x from the set S;
```

Figure 17. The Character-Shape Covering Algorithm (CSC).

To seek alternative pangrams for the unligative dataset, we called for an online competition on character-shape pangram composition. Our semi-automatic GUI editor was provided to competitors. The texts were evaluated for the pangram condition, lipogram condition and compactness. The winner text is shown in Figure 18. It contains 851 characters from Arabic proverbs and clichés. It covers character-shapes so that only

omni-ligatives are not asserted to be unligative. We feel it can be made more compact for future datasets.

أضيق الأمر أدناه من الفرج	احذر صديقك ألف مرة	اختلط حابلهم بنابلهم
الإفراط في التواضع يجلب المذلة	الشجاع من لا يغضب	رأى الناس الهلال
احذروا ما قرب من الارض	احفظ قرشك الأبيض ليومك الأسود	إذا أتت مذمتي من ناقص فهي الشهادة لي بأني
إن البغاث بأرضنا يستنسر	يُحسن الجار لجاره	كامل
الذل في الحرص وفي الوضاعة	اكثر قراءة القرآن	يشفع الرسول لأمته
بقدر الرأي تعتبر الرجال وبالآمال ينتظر المآل	الأخ لأخيه	أبلغ الرأي نصيحة حازم
تكاثرت الظباء على خراش فما يدري خراش ما يصيد	أخاك أخاك فإن مَنْ لا أخا له كَساعٍ إلى	إذا تفرقت الغنم قادتها العنز الجرباء
تناس مساوئ الإخوان يدم لك وُدُّهُمْ	الهيجا بغير سلاح	إذا تكلمت بالكلمة ملكتك
الحق أبْلَجُ والباطل لجلج	لا تسيء ظنونك	إذا تمنيت فاستكثر
الاعين موانئ	و إن سئلت لا تسوّف	إذا ذكرت الذئب فأعد له العصا
تطبَخ الام الطعام	إذا فرغ الفؤاد ذهب الرقاد	إبرة في كومة قش
سلامة العقل قبل سلامة الالفاظ	إذا قل ماء الوجه قل حياؤه	اتسع الخرق على الراقع
السلاح ثم الكفاح	إذا هَبَّتْ رياحك فاغتنمها	اجلس حيث يُؤْخَذُ بيدك وَتُبَرُّ ولا تجلس حيث
شعيرنا ولا قمح غيرنا	تجلَّدِ و اعلم بأن الدهر غير مُخَلَّدِ	يؤخذ برجلك وتُجَرُّ.

Figure 18. An Arabic character-shape pangram, composed from proverbs and clichés, with the lipogram condition.

Compactness of the Unligative Text and The Isolated Letters Parts

In this subsection, we study the character-bound (CB) and the PAW-bound (PB) for two hypothetical PAW-based unligative datasets. The two datasets derive from extreme assumptions on the level of ligativity of an alphabet (or font), viz. the high-ligativity (HL) and the low-ligativity (LL) assumptions. High-ligativity assumes that all character-shapes are omni-ligative except for one (B), one (M) and one (E) instances. Low-ligativity depicts a case where a distinct unligative character-shape can be found for a set of PAWs that form a pangram. Low-ligativity can become more probable if character-shapes that ligate frequently are used earlier in the CSC algorithm.

The HL and the LL assumptions lead to worst and best-case scenarios with respect to CB and PB, regardless of the underlying alphabet. The following observations facilitate the derivation of CB and PB for the HL and LL assumptions. Denote the number of (A), (B), (M) and (E) character-shapes in a given model by |A|, |B|, |M| and |E|, respectively. Then,

- (A) character-shapes may appear only as a single character PAWs. Hence, CB and PB equations must include one |A| term.

- (B) and (E) character-shapes appear exactly once per PAW.

- (M) character-shapes can only exist within PAWs of at least three characters.

- |E| is larger than |M| and |B| in all typographic models, as revealed by Table 11. Hence, we need at least |E| PAWs to include all (E) shapes in the dataset, repeating some (B) character-shapes.

Equation (3.1) formulates PB under the LL assumption. In addition to the |A| single-character PAWs, we need as many multi-character PAWs as the maximum of |B| and |E|.

$$PB_{LL} = |A| + MAX(|B|,|E|) = |A| + |E| \quad (3.1)$$

Equation (3.2) derives a CB expression for the LL assumption from Equation (3.1).

$$CB_{LL} = |A| + |M| + 2*MAX(|B|,|E|) = |A| + |M| + 2*|E| \quad (3.2)$$

The 2 * MAX(|B|,|E|) + |M| terms of Equation (3.2) account for the minimum number of characters in MAX(|B|,|E|) PAWs that may include up to |M| character-shapes.

Equation (3.3) reveals that PB under the HL assumption is of the order of the total count of character-shapes.

$$PB_{HL} = |A| + |E| + |B| + |M| - 2 \quad (3.3)$$

|B| PAWs are needed for all (B) character-shapes to appear with their unique unligative neighbor. Similarly, |M|-1 and |E|-1 additional PAWs are needed to cover the (M) and (E) character-shapes with their respective unligative neighbors. The 1 is subtracted in order to avoid double-counts of the unligative placeholder character-shapes.

Equation (3.4) maps PB of Equation (3.3) into CB.

$$CB_{HL} = |A| + 2 * |E| + 2 * |B| + 3 * |M| - 4 \quad (3.4)$$

In the best-case scenario, the |B| and |E| PAWs of Equation (3.3) are bigrams that contribute 2 * |B| and 2 * |E| character-shapes, respectively. (M) character-shapes may appear in PAWs of length 3 or more. Assuming ternary PAWs are used, 3 * |M| character-shapes are needed to make |M| PAWs. We subtract 4 from the sum to account for repetitions of character-shapes used as placeholders.

Table 16 shows character and PAW bounds under the LL and HL assumptions for the typographic models of Table 10.

Table 16. Character and PAW bounds under the low-ligativity and the high-ligativity assumptions.

Model	PB_{LL}	CB_{LL}	PB_{HL}	CB_{HL}
Traditional	71	115	129	217
Dot-Less	37	57	66	106
2-Shapes	40	64	78	126
Dot-Less & 2-Shapes	21	32	41	63

Further reductions in the size of the ligative dataset can benefit from linguistic analysis. For instance, ancient Arabic cryptanalysts Al-Kindi, Ibn Dunaineer and Ibn Duraihem [123] had compiled lists of characters sequences that cannot appear in a given

word. In Table 17, we display 67 ligative bigrams of such stop-list. Discarding some or all of these reduces the size of dataset forms. The stop-list can also be applied in error detection for text recognition [124], [125].

Table 17. Linguistically excluded ligatures according to Arabic cryptanalysis.

Al-Kindi	MM	ME	BM	BE
	ظجـ	ـظج	ظجـ	ظج
	ظحـ	ـظح	ظحـ	ظح
	ظخـ	ـظخ	ظخـ	ظخ
	صجـ	ـصج	صجـ	صج
		ـصز		صز

Ibn Dunaineer	MM	ME	BM	BE
		ـسز		سز
		ـثز		
	حخـ	ـحخ	حخـ	حخ
	خحـ	ـخح	خحـ	خح
	عحـ	ـعح	عحـ	عح
	غحـ	ـغح	غحـ	غح
	غخـ	ـغخ	غخـ	غخ
	قجـ	ـقج	قجـ	قج
	طجـ	ـطج	طجـ	طج
	غجـ	ـغج	غجـ	غج

Ibn Duraihem	MM	ME	BM	BE
		ـضز		ضز
	هجـ	ـهج	هجـ	هج
	هخـ	ـهخ	هخـ	هخ
	عخـ	ـعخ	عخـ	عخ

3.2.3 The Passages Part of Dataset

The passages PoD aims at having a distribution of character-shapes near to natural. Natural distribution of a dataset has tremendous advantages in training and testing. Training on data that is abundant in natural language should improve the system on such data; hence reduces the overall error. On the other hand, testing on near-to-natural data distributions gives clearer insight to real life error rates.

The passages PoD consists of semi-automatically selected news text from the Gigaword corpus [126]. Texts of around 50 words long are automatically chunked from the corpus. A human reader then asserts that the content of the paragraphs is suitable for the dataset forms. Probabilities of character-shapes of the selected paragraphs are compared to those estimated from the Gigaword corpus. If they don't match, some paragraphs are replaced by more representative ones. The dataset, as a whole, should ensure a level of natural distribution of character-shapes, but without guarantee on larger

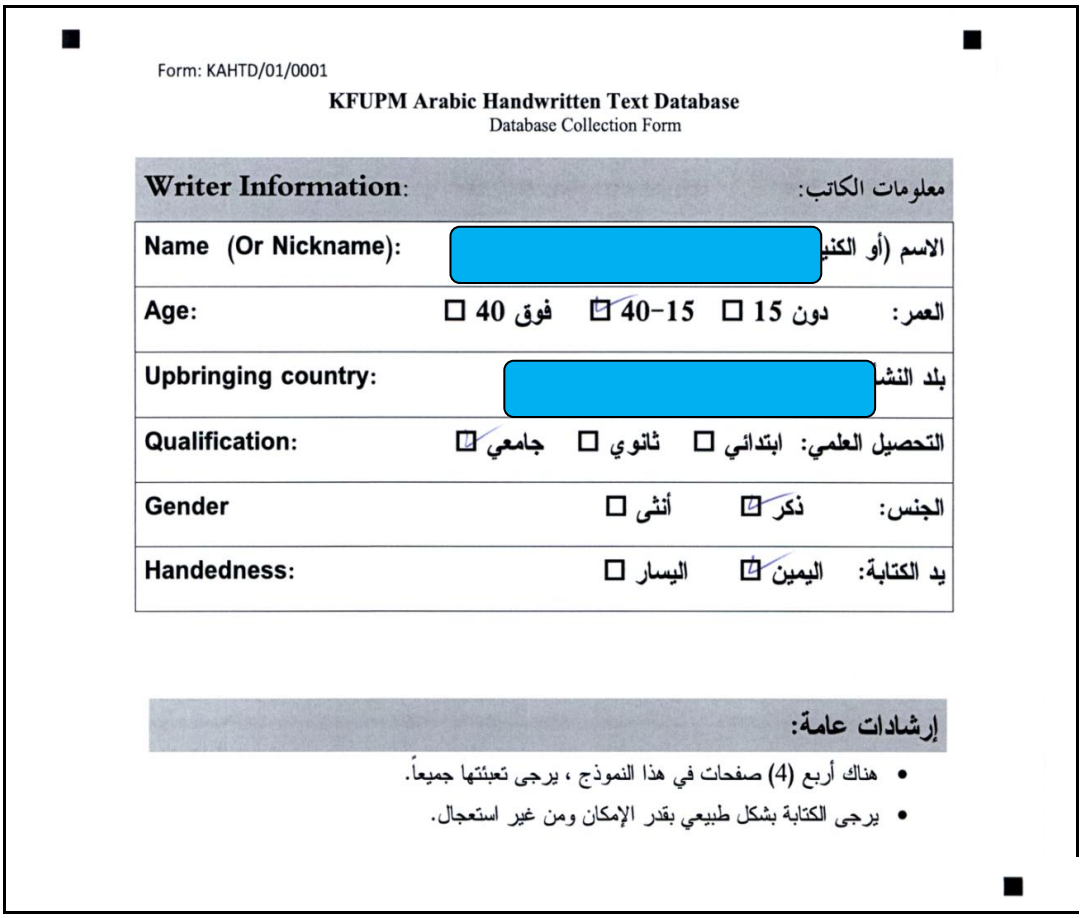
n-grams. The character-shape probabilities are shown side-to-side with the corresponding Gigaword [126] probabilities in Appendix E.

3.2.4 The Repeated Phrases Part of Dataset

The repeated phrases part consists of a set of commonly used phrases that are to be written six repeated times per form. This part is the only part where the distribution of the covered units per form uniformly goes above one. It is designed with writer identification research in mind.

3.3 Form Collection

A *form* is an instance of the dataset intended to be filled by a single writer. Each form contains four pages. A four-paged sample form is shown in Figure 19 to Figure 22. Each form contains ligative and unligative parts and is intended to be filled by a distinct writer. The forms were printed and distributed mainly among the community in King Fahd University of Petroleum & Minerals. After discarding incomplete forms, 450 forms were selected. More than half of the forms were scanned into TIFF colored images with a resolution of 300 dpi.



Form: KAHTD/01/0001

KFUPM Arabic Handwritten Text Database

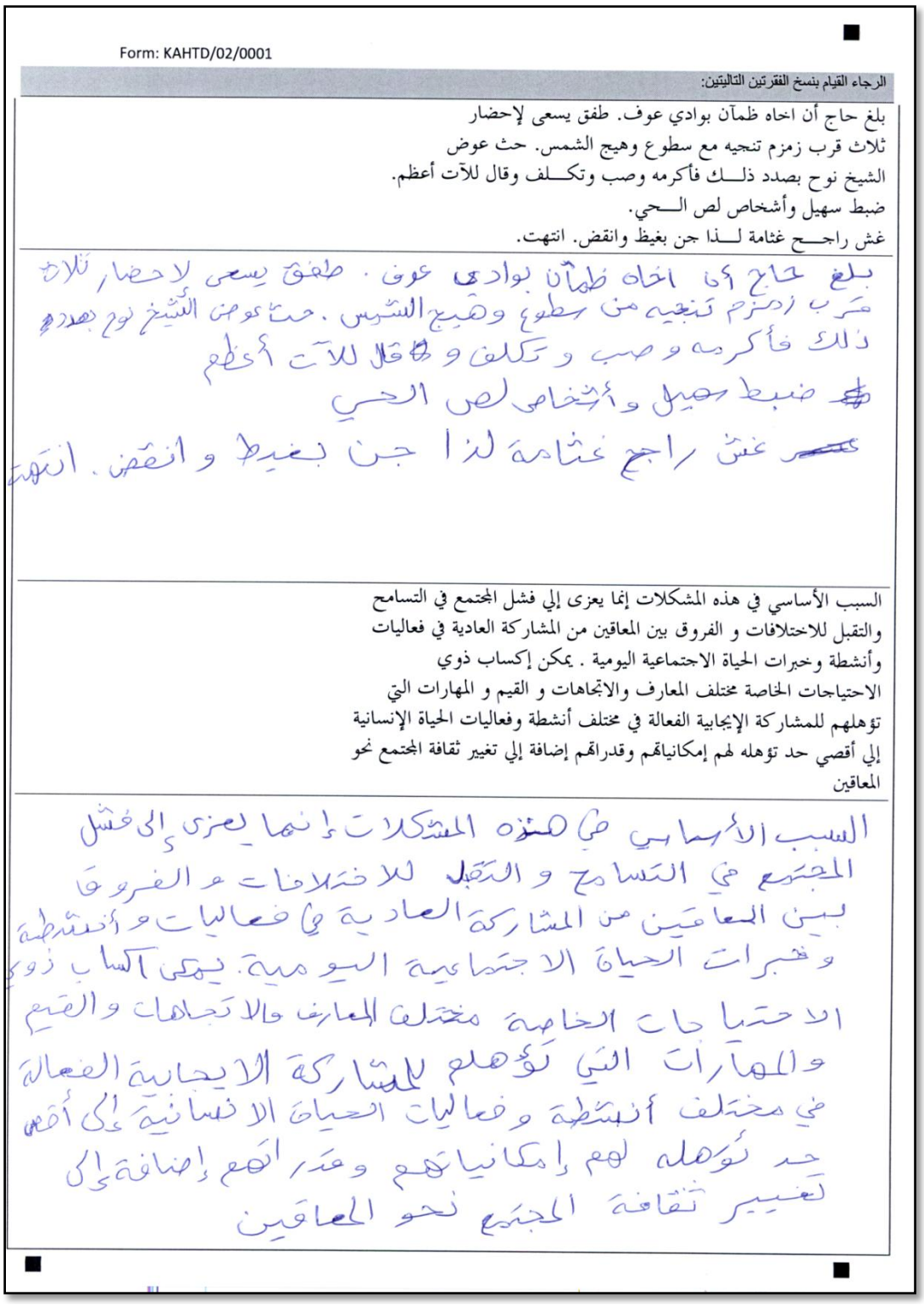
Database Collection Form

Writer Information:	معلومات الكاتب:
Name (Or Nickname):	الاسم (أو الكنية):
Age:	العمر: ☐ دون 15 ☑ 15-40 ☐ فوق 40
Upbringing country:	بلد النشأة:
Qualification:	التحصيل العلمي: ☐ ابتدائي ☐ ثانوي ☑ جامعي
Gender	الجنس: ☑ ذكر ☐ أنثى
Handedness:	يد الكتابة: ☑ اليمين ☐ اليسار

إرشادات عامة:

- هناك أربع (4) صفحات في هذا النموذج ، يرجى تعبئتها جميعاً.
- يرجى الكتابة بشكل طبيعي بقدر الإمكان ومن غير استعجال.

Figure 19. A scanned sample of the first form-collection page where writers' information is filled.


Form: KAHTD/02/0001

الرجاء القيام بنسخ الفقرتين التاليتين:

بلغ حاج أن اخاه ظمآن بوادي عوف. طفق يسعى لإحضار
ثلاث قرب زمزم تنجيه مع سطوع وهيج الشمس. حث عوض
الشيخ نوح بصدد ذلـــك فأكرمه وصب وتكـــلف وقال للآت أعظم.
ضبط سهيل وأشخاص لص الـــحي.
غش راجـــح غثامة لـــذا جن بغيظ وانقض. انتهت.

بلغ حاج أن اخاه ظمآن بوادي عوف. طفق يسعى لاحضار ثلاث
قرب زمزم تنجيه من سطوع وهيج الشمس. حث عوض الشيخ نوح بصدد
ذلك فأكرمه وصب و تكلف و قال للآت أعظم
ضبط سهيل و أشخاص لص الحي
غش راجح غثامة لذا جن بغيظ و انقض. انتهت

السبب الأساسي في هذه المشكلات إنما يعزى إلي فشل المجتمع في التسامح
والتقبل للاختلافات و الفروق بين المعاقين من المشاركة العادية في فعاليات
وأنشطة وخبرات الحياة الاجتماعية اليومية . يمكن إكساب ذوي
الاحتياجات الخاصة مختلف المعارف والاتجاهات و القيم و المهارات التي
تؤهلهم للمشاركة الإيجابية الفعالة في مختلف أنشطة وفعاليات الحياة الإنسانية
إلي أقصي حد تؤهله لهم إمكانياتهم وقدراتهم إضافة إلي تغيير ثقافة المجتمع نحو
المعاقين

السبب الأساسي في هذه المشكلات إنما يعزى إلى فشل
المجتمع في التسامح و التقبل للاختلافات و الفروق
بين المعاقين من المشاركة العادية في فعاليات و أنشطة
وخبرات الحياة الاجتماعية اليومية. يمكن اكساب ذوي
الاحتياجات الخاصة مختلف المعارف والاتجاهات والقيم
والمهارات التي تؤهلهم للمشاركة الايجابية الفعالة
في مختلف أنشطة وفعاليات الحياة الانسانية إلى أقصى
حد تؤهله لهم إمكانياتهم وقدراتهم إضافة إلى
تغيير ثقافة المجتمع نحو المعاقين

Figure 20. A scanned sample of the second form- collection page where the unligative text part (top) and the natural statistics part (bottom) are filled.

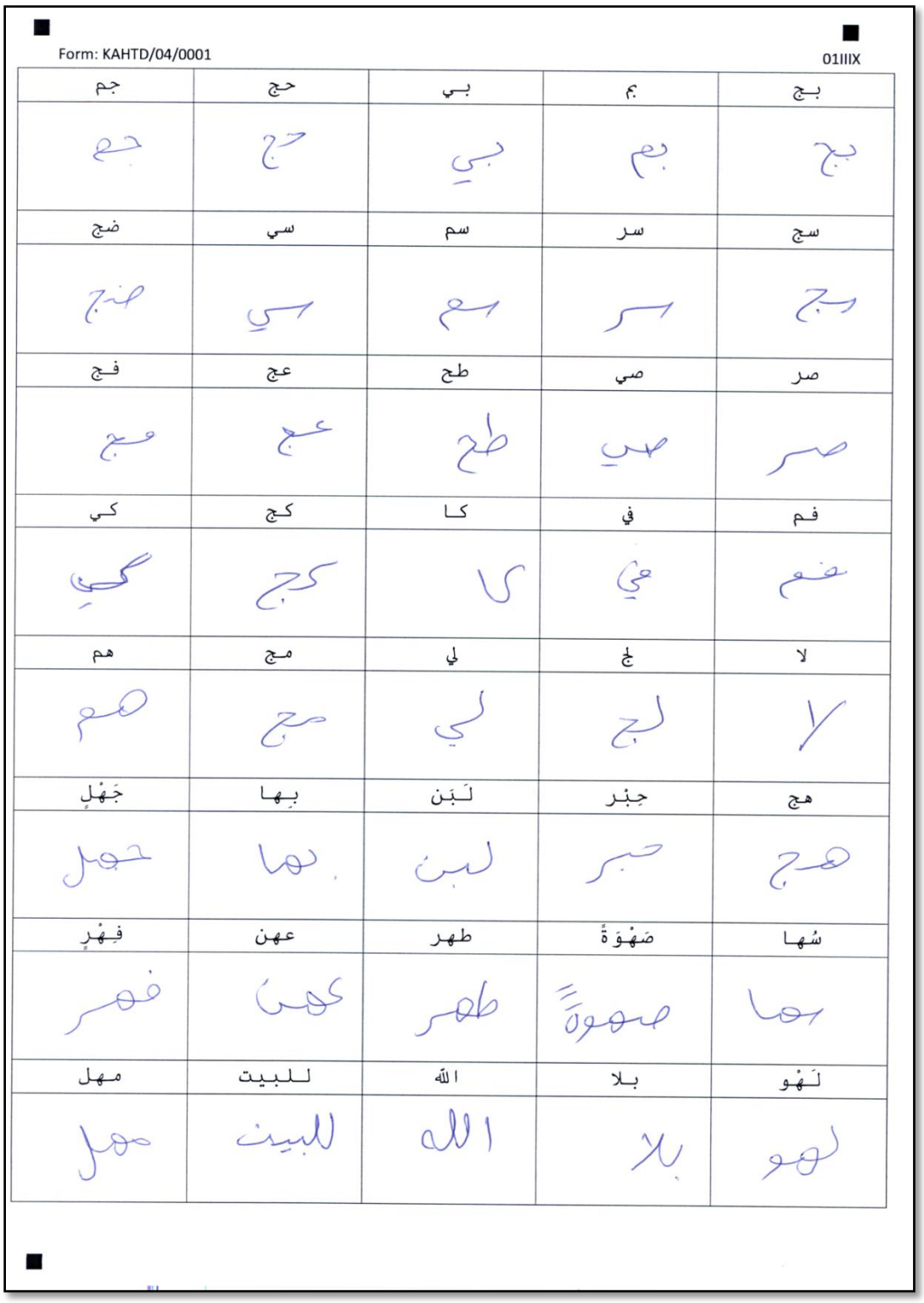


Figure 21. A scanned sample of the third form-collection page concerned with the ligatures part.

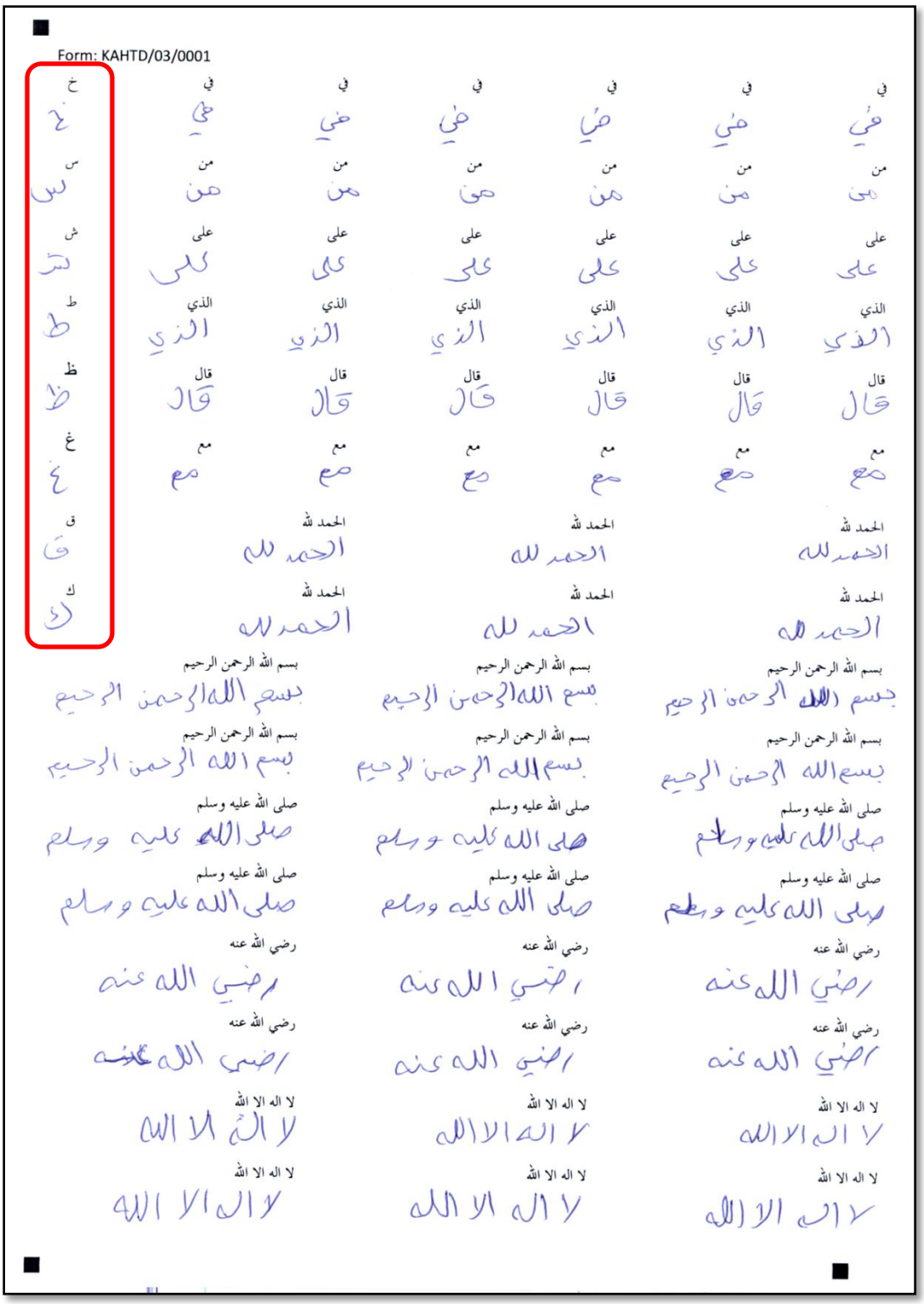


Figure 22. A scanned sample of the fourth form-collection page. The isolated characters part is marked in a box. The rest collects repeated words and phrases.

Statistics of the regions, genders, writing-hands, and qualifications of the writers are presented in Table 18, where we consider the region of the writer as one of the following three:

- the Arab Peninsula: containing the Gulf countries, Yemen and Iraq,
- North Africa: containing Egypt, Sudan, and the countries of Northwest Africa, and
- Levant: containing Syria, Jordan, Palestine, and Lebanon

The designs of the ligative and unligative parts of the forms are discussed in Sections 5.1 and 5.2, respectively.

Table 18. Numbers of writers in the collected dataset per region, gender, handedness and qualifications.

Region	Arab Peninsula	417
	North Africa	22
	Levant	11
Gender	Male	398
	Female	52
Writing Hand	Right Hand	416
	Left Hand	34
Qualification	Intermediate School	4
	High School	386
	B.Sc. / BA	53
	M.Sc. / Ph.D.	7

The forms of the ligative dataset are designed to accommodate 40 words/PAWs in single-paged grids like the ones shown in Figure 23. Each form covers ligatives under the combined reduction-model. Collectively, twelve distinct forms were needed to cover the 2-Shapes model. This is achieved by making forms that contain the different dot-less representatives of each ligative, as shown in Table 19. The last three columns of the table contain some unligative patterns for other objectives.

Figure 23. Examples of ligative forms.

Table 19. Final design of 12 distinct but related forms with 40 entries each.

Form	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	بج	بم	بي	حج	جم	سج	سر	سم	سي	ضج	صر	صي	طح	عج	فج	فم	في	كا	كج	كي	لا	لج	لي	مج	هم	هج
2	تج	تم	تي	خج	حم	شج	شر	شم	شي	صح	ضر	ضي	طخ	عخ	فح	قم	قي	كإ	كح	كئ	لإ	لح	لئ	مح	مم	هح
3	ثج	ثم	ثي	جح	خم	سح	شز	صم	سئ	ضح	ضز	صئ	طم	عخ	قح	عم	فئ	كأ	كخ	كى	لأ	لخ	لى	مخ	لم	هخ
4	نج	نم	ني	حح	جم	شح	سر	ضم	شئ	صخ	صر	ضئ	ظم	عج	فخ	غم	قئ	كآ	كج	كي	لآ	لج	لي	مج	كم	هج
5	يج	يم	يي	جخ	حم	سخ	شر	سم	سى	ضخ	ضر	صى	طح	عخ	قخ	فم	فى	كك	كح	كئ	لا	لح	لئ	مح	هم	هح
6	ئج	ئم	ئي	خخ	خم	شخ	شز	شم	شى	بخ	ضز	ضى	طخ	عخ	جج	قم	قى	كل	كخ	كى	لإ	لخ	لى	مخ	مم	هخ
7	بح	بم	بئ	حج	جم	سج	سر	صم	سي	تخ	صر	صي	طم	عج	فج	عم	في	كا	كج	كي	لأ	لج	لي	مج	لم	هج
8	تح	تم	تئ	خج	حم	شج	شر	ضم	شي	ثخ	ضر	ضي	ظم	عخ	فح	غم	قي	كإ	كح	كئ	لآ	لح	لئ	مح	كم	هح
9	ثح	ثم	ثئ	جح	خم	سح	شز	سم	سئ	نخ	ضز	صئ	طح	عخ	قح	فم	فئ	كأ	كخ	كى	لا	لخ	لى	مخ	هم	هخ
10	نح	نم	نئ	حح	جم	شح	سر	شم	شئ	يخ	صر	ضئ	طخ	عج	فخ	قم	قئ	كآ	كج	كي	لإ	لج	لي	مج	مم	هج
11	يح	يم	يئ	جخ	حم	سخ	شر	صم	سى	ئخ	ضر	صى	طم	عخ	قخ	عم	فى	كك	كح	كئ	لأ	لح	لئ	مح	لم	هح
12	ئح	ئم	ئئ	خخ	خم	شخ	شز	ضم	شى	بنج	ضز	ضى	ظم	عخ	جج	غم	قى	كل	كخ	كى	لا	لخ	لى	مخ	كم	هخ

Form	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	جِبْر	لَبَن	بِها	جَهْلٍ	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	لَهْو	بلا	الله	للبيت	مهل
2	ستر	فِتَنٌ	ثَهَبُ	شَرْحُها	شَهِيٌّ	أرضها	ظهر	بَلَغَها	قَهْر	كَهْل	لِلإِنْسِ	محمد	بُص	صوف
3	نثر	بَحَثْنَ	لَبِسَتْها	فَرْخُها	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	ههنا	مَلَأ	ينبسط	بَط	طن
4	كَنَر	مَنَنٍ	نَهار	جَهْلٍ	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	مها	لِلآن	بسط	بِع	بعد
5	خَيْر	تين	يَهَبُ	شَرْحُها	شَهِيٌّ	أرضها	ظهر	بَلَغَها	قَهْر	لَهْو	بلا	عقل	بَقّ	يعسر
6	بِئر	يَئِنُّ	نَبَتَها	فَرْخُها	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	كَهْل	لِلإِنْسِ	لقط	السنية	جل
7	جِبْر	لَبَن	بِها	جَهْلٍ	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	ههنا	مَلَأ	نبه	دجاجك	تونس
8	يَتَزَكّى	فِتَنٌ	ثَهَبُ	شَرْحُها	شَهِيٌّ	أرضها	ظهر	بَلَغَها	قَهْر	مها	لِلآن	عربي	جد	حقٌّ
9	ثِيثِرْ	بَحَثْنَ	لَبِسَتْها	فَرْخُها	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	لَهْو	بلا	نبي	جِس	حور
10	كَنْز	مَنَنٍ	نَهار	جَهْلٍ	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	كَهْل	لِلإِنْسِ	يتم	كمال	علي
11	ضِيزَى	تين	يَهَبُ	شَرْحُها	شَهِيٌّ	أرضها	ظهر	بَلَغَها	قَهْر	ههنا	مَلَأ	نبها	خط	كفا
12	يَشْمَئِزُّ	يَئِنُّ	نَبَتَها	فَرْخُها	سُها	صَهْوَةً	طهر	عهن	فِهْرٍ	مها	لِلآن	بيت	جُع	ماكر

Table 19 shows ligative per form. Each form contains approximately 120 character-shapes. Table 20 summarizes coverage information about the parts of the collected dataset.

Table 20. Units and coverage criteria and designs of the different forms of the dataset.

Parts	Covering Units	Covered Unit Model	Coverer	Frequency of the covered unit	Figure Number
Unligative text (UT)	Paragraphs	all character-shapes	form	Uniform (1 per form)	Figure 20 (Top)
Isolated characters (IL)	Isolated Characters				Figure 22 (Left-Top)
Passages	Paragraphs	all character-shapes	dataset	Natural per dataset	Figure 20 (bottom)
Ligatures	Isolated ligatures and words	dot-less, 1-shape	form	Uniform (almost 1 per form)	Figure 21
		1-shape ligatures	12 forms	Uniform (less than 1 per form)	
Repeated phrases	Words and sentences	Selected words/sentences	form	Uniform (more than one per form)	Figure 22

3.4 Data Preparation

Pages of the dataset forms are scanned at a resolution of 300 dpi. The scanned images undergo preprocessing steps.

3.4.1 Form-Page Deskew and Classification

To ease skew detection and correction (deskew) and to ease page classification of the forms pages, three aligned black boxes are printed on the corners of each page. The boxes are printed in positions so that if their centers of gravity are connected, a right angle with sides parallel to the original reference coordinates of the page is formed. The skew angle θ, taken between the current, say x-, axis and the corresponding original axis can be estimated from the arctangent equation:

$$\tan^{-1}\theta = \frac{y2-y1}{x2-x1} \quad (3.5)$$

where (x1, y1) and (x2, y2) are the centers of gravity of the two boxes on the short side of the scanned image. Deskew is done by rotating the image in the direction of - θ.

The black boxes of the pages are automatically recognized by conditioning the area- and aspect ratio- (height/width) features of all foreground objects against pre-set thresholds. Box positions help in classifying a page into one of the 4 pages a form can have. Each page category has the head of the right angle formed by the three black boxes at a different corner of the page.

3.4.2 Block of Handwriting Extraction and File Naming

The extraction of blocks of handwriting (BoHs) from form pages is eased by providing boxes for the printed text and for the handwriting. For all except Page 3 of the forms, the boxes were printed on the front of the form pages. For every third page, the frames were printed on the back side of the page so that its shadow appears when scanning. This was suggested to avoid constraining writers with boxes. By knowledge of the page structure, specialized tools to extract BoHs are implemented.

For example the ligatures part has a grid that can be recognized as the biggest foreground object in terms of height and width Figure 24. Borders of each box in the grid are defined by high horizontal and vertical profile values (detailed next chapter). Vertically, printed and handwritten texts appear alternating in every column. Column, row and form numbers are used to identify the correspondent GT of each handwritten ligature to provide adequate naming. In Chapter 4, we discuss cases where enclosing boxes contain BoHs that need to be segmented and named to the character-shape level.

(a) **(b)**

Figure 24. A sample of the ligatures part (a) and its grid (b)

CHAPTER 4

SEGMENTATION AND GROUND-TRUTHING

The blocks of handwriting (BoHs) extracted from the Unligative Text (UT) part of the dataset (PoD) need to be segmented into character-shapes and aligned with the corresponding ground-truth (GT). The segmentation and alignment process is often called ground-truthing (GTing). GTing is usually a semi-automatic process. Pixel-level GTs assign a distinct label to all pixels that contribute to a unit in the image. This is not to be confused with character-level GTs, where the image and the corresponding text are provided without character-shape-distinction in the image.

The ultimate level of text ground-truthing is the character level. Character-level ground-truths associate a distinct label to the pixels that correspond to a character in a document image. They provide a powerful resource for the development of character segmentation and recognition systems [127]. However, character-level ground-truths are scarce, mainly because of the human efforts and time required to generate them. The dataset of the University of Washington, UW-III, for example, has 979 document images with known text, but only 33 of them contain character-level ground-truths [128].

Character-level ground-truths and character segmentation algorithms engage in a "chicken and egg" relationship. If characters were segmented, obtaining automatic ground-truths would have become an easier task, and if character-level ground-truths were available, the evaluation of character segmentation algorithms would have been easy. One way to break this recursion is by human intervention. Fortunately, most writing

occurs with some spatial and temporal sequence and Arabic is not an exception [129]. Hence, character-level ground-truthing can be performed by only identifying the borders of each character [130]. Semi-automatic tools that determine such borders can be deployed to ease the task [131].

Hereafter, we reserve the term *segmentation* to the automatic labeling of characters in handwriting and the term *ground-truthing* to their human-guided labeling. Segmentation can be performed either blindly or non-blindly. Blind segmentation relies solely on information from an image to label text components. Non-blind segmentation, also known as text-alignment, exploits information about the corresponding text. Figure 25 shows the block diagrams for ground-truthing, blind-segmentation, and non-blind segmentation.

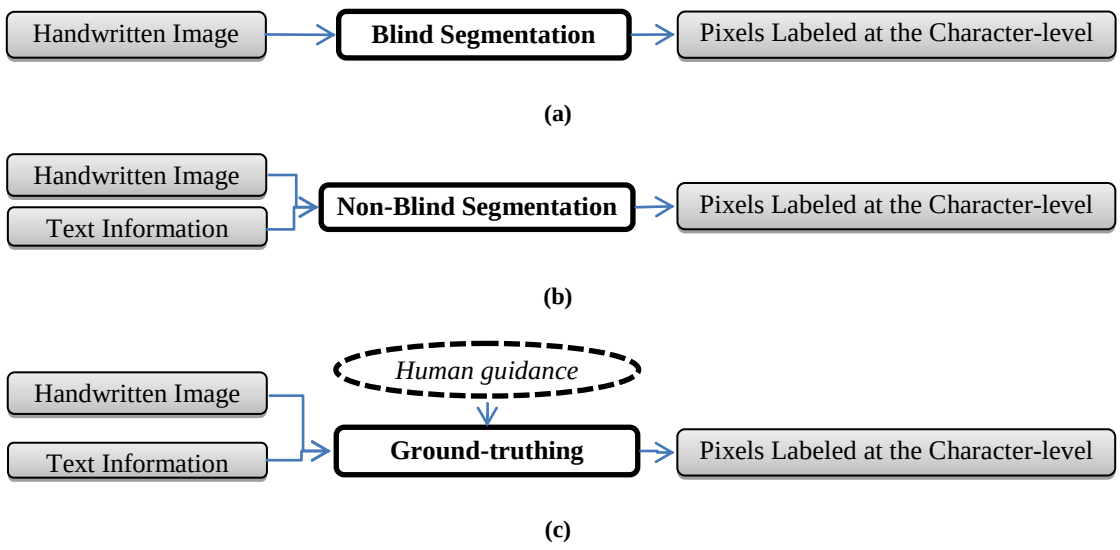


Figure 25. Block diagrams of (a) blind segmentation, (b) non-blind segmentation, and (c) ground-truthing.

We present different approaches to GT, segment and align the UT PoD. We start by GTing for the following twofold benefits: GTs can be used in segmentation evaluation; and they provide clean inputs to synthesis algorithms to prevent error

propagation. We report line segmentation and intents to blindly segment lines into character-shapes. Since lines are not aligned with GT, non-blind segmentation is not a choice. We use GTed words to make a word-dataset from the current UT PoD and apply non-blind segmentation and alignment algorithms on it. We finally introduce a new entropy-based evaluation method for Arabic segmentation from words to PAWs and to character-shapes. Figure 26 shows Arabic samples from the UT PoD ground-truthed at the levels of text-lines words, pieces of Arabic word (PAW) and characters.

(a) **(b)** **(c)** **(d)**

Figure 26. Ground-truths at the (a) text-line-level, (b) word-level, (c) PAW-level and (d) character-level.

Arabic character segmentation is an open research problem [132]; especially that Arabic script is inherently cursive and that PAWs may vertically overlap. Lack of character-level benchmarks and objective evaluation methods are among the most important causes for the tardiness of the solution to the character segmentation problem in Arabic. In this chapter, we provide automatic and semi-automatic character-level ground-truthing for Arabic characters. We also introduce a quantitative evaluation method for Arabic handwriting segmentation.

4.1 Preprocessing and Common Tools

The BoHs extracted from the UT PoD undergo some conventional conversions from colored space to binary space passing through the gray-level space. Connected

components (*aka* blobs) and projections of the binary images are then prepared to be used in later stages. Blobs that are smaller than pre-specified height and width thresholds are filtered out as noise. Deskew is performed on the extracted BoHs and then repeated on single lines.

4.1.1 Projections

An image projection, or profile, maps the 2-D space of a binary image into a numerical vector. The vertical projection (VP) assigns the number of foreground-colored pixels of a column to an entry in the output vector that corresponds to that column. Similarly, the horizontal projection (HP) generates a vector of the same size of the image height where each entry contains the count of foreground-colored pixels of the row that they correspond to. Some of our algorithms rely on vertical and horizontal projections; however, we use a *smoothed* version of the projection profiles.

The smoothed projection assigns the average number of foreground-colored pixels in *m* consecutive columns/rows to the output entry corresponding to their centers. At the image borders, zero-padding is assumed. Equations (4.1) and (4.2) represent the smoothed vertical and horizontal projections, respectively.

$$\mathbf{VP^m[c]} = \sum_{\mathbf{cols=c-\lfloor \frac{m}{2} \rfloor}}^{\mathbf{c+\lfloor \frac{m}{2} \rfloor}} \sum_{\mathbf{r=1}}^{\mathbf{img\ height}} \mathbf{img(r, cols)} \Big/ \mathbf{m} \qquad (4.1)$$

$$\mathbf{HP^m[r]} = \sum_{\mathbf{rows=r-\lfloor \frac{m}{2} \rfloor}}^{\mathbf{r+\lfloor \frac{m}{2} \rfloor}} \sum_{\mathbf{c=1}}^{\mathbf{img\ width}} \mathbf{img(rows, c)} \Big/ \mathbf{m} \qquad (4.2)$$

where *img(x,y)* denotes the value of a pixel at Row *x* and Column *y*, which is 1 if it has the foreground color and 0 otherwise, and ⌊ ⌋ denotes the truncation operation.

For simplicity, the smoothed vertical and horizontal projections are hereafter referred to as VP and HP, respectively, whenever *m*, *c* and *r* do not need to be specified.

4.1.2 Block of Handwriting Deskew

Block of handwriting deskew simplifies line segmentation. BoH deskew aims at maximizing the sum of the squares of the horizontal projection values (HP) of the BoH. It empirically tries a set of angles around zero and selects the HP-maximizing one. Algorithm Deskew is a classical algorithm described in [133] and outlined in Figure 27.

Algorithm: Deskew

Inputs:

- Image ***img***, containing a BoH
- Integer ***m***, the averaging factor

Output: Deskewed Image ***imgDeskew***

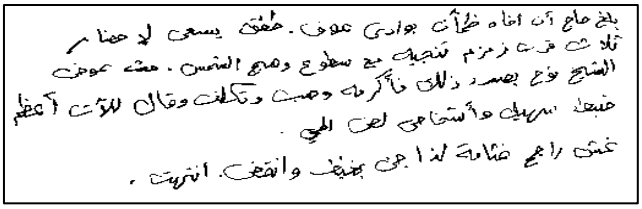
1. θ = 0
2. MAXSquareSum = $\sum_r (\mathrm{HP}_{img}{}^m\{\mathrm{r}\})^2$
3. ***for*** θ in range from $-|\theta_1|$ to $|\theta_2|$
   - 3.1. ***imgD*** = img rotated by angle θ
   - 3.2. Obtain $\mathrm{HP}_{imgD}{}^m\{\mathrm{r}\}$ from ***imgD***
   - 3.3. SquareSum = $\sum_r (\mathrm{HP}_{imgD}{}^m\{\mathrm{r}\})^2$
   - 3.4. ***if*** SquareSum > MAXSquareSum
     - 3.4.1. MAXSquareSum = SquareSum
     - 3.4.2. ***imgDeskew =*** *imgD*
     - ***end if***

end for

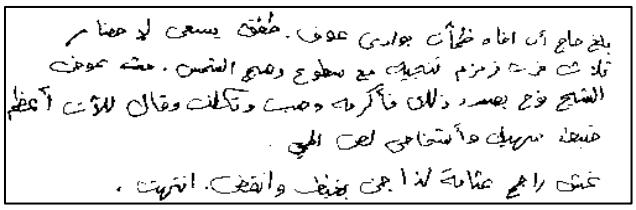
end Algorithm **Deskew**

Figure 27. Deskew algorithm.

The algorithm favors lengthy horizontal text-lines over shorter ones due to the squaring operation. This is adequate to the Arabic script where the abundance of horizontal Kashida is ideally high. Skew correction rotates the image line with accordance to the chosen angle. Figure 28 shows a paragraph before and after block skew correction. Visual inspection revealed no failures when the range was defined from -5^o to 5^o with a step of 0.1^o.



(a)



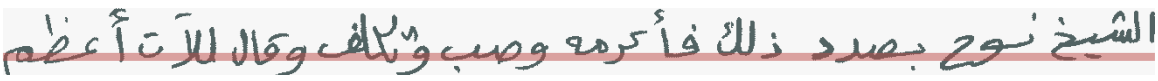
(b)

Figure 28. A sample paragraph (a) before and (b) after global deskew correction

4.1.3 Baseline Estimation

The term baseline (BL) is used here to refer to the range of rows containing the row with the highest foreground pixel count and all its neighbor rows with pixel counts above a fraction, ***factor***, of the maximum pixel count. BL is usually computed on a word or chunk of words in a single line. The chunks should neither be too short nor too long. Figure 29 (a) shows an example of BL miss-estimation in a chunk that is of 3 character-shapes only. The BL range surrounds the descendent of a character-shape that happens to exceed in HP the Kashida level. This would be less likely if the text had contained a representative amount of character-shapes. Figure 29 (b) shows a case where BLs on chunks would be more accurate than a unified BL, due to a wavy writing style over and under the estimated BL.

(a)



(b)

Figure 29. Baseline miss-estimation for (a) a short line and for (b) a long wavy line.

We present two baseline-range estimation algorithms: the Single Baseline-Range Estimation algorithm (SBRE) that estimates a single baseline-range for a text-line image and the Multiple Baseline-Range Estimation algorithm (MBRE) that estimates localized baseline-ranges. SBRE is simpler and needs fewer inputs while MBRE is more complex but adequate for wavy long text-lines.

The SBRE algorithm, listed in Figure 30, assigns as a baseline-range the maximal set of contiguous rows containing the row with maximum HP but no rows with HP values less than a specified factor of it.

Algorithm: Single Baseline-Range Estimation (SBRE)

Inputs:

- Array ***HP{r}*** of the smoothed horizontal projection of an image
- Fraction ***factor*** with a value between *0* and *1*

Output: ***Pair of Integers*** **(*u, d*)** representing the upper and lower y-coordinates of the baseline-range

Steps:

1. Let h_{max} be the **maximum value** in **HP{r}**; *i.e.* h_{max} **= Max(*HP{r}*)**
2. Let h_r be the **row-number** of h_{max} ; *i.e.* h_r **= ArgMax(*HP{r}*)**
3. ***bl*** = ***factor**** h_{max}
4. Set ***u*** to be the nearest row-number to h_r satisfying the following conditions:
   - 4.1. ***u*** is above h_r
   - 4.2. ***HP{u} ≥ bl***
   - 4.3. ***HP{****the row above* ***u} < bl***
5. Set ***d*** to be the nearest row-number to h_r satisfying the following conditions:
   - 5.1. ***d*** is below h_r
   - 5.2. ***HP{d} ≥ bl***
   - 5.3. ***HP{****the row below* ***d} < bl***

end Algorithm **Single Baseline-Range Estimation (SBRE)**

Figure 30. Listing of the SBRE algorithm

The MBRE algorithm, listed in Figure 31, assigns a baseline-range to different segments. Text-line segments are horizontal partitions of the text-line image. MBRE defines them as maximal contiguous parts of the text-line image where VP is greater than or equal to a certain threshold. Segments that are above-average in width are assigned their baseline-ranges via SBRE. Narrower segments are assigned interpolated values

according to the nearest right and left segments with assigned ranges. The interpolated ranges may differ according to their order of computation; we adhere to computing them from right to left.

Algorithm: Multiple Baseline-Range Estimation (MBRE)

Inputs:

- Array ***HP{r}*** containing the m-averaged horizontal projection of the text-line image,
- Array ***VP{c}*** containing the m-averaged vertical projection of the text-line image,
- Fraction ***factor,*** a baseline factor
- Integer ***cut,*** a threshold on VP
- Number ***Threshold***

Output: An Array of **ordered pairs of integers** (u_i, d_i) representing the y-coordinates of the baseline-range

Steps:

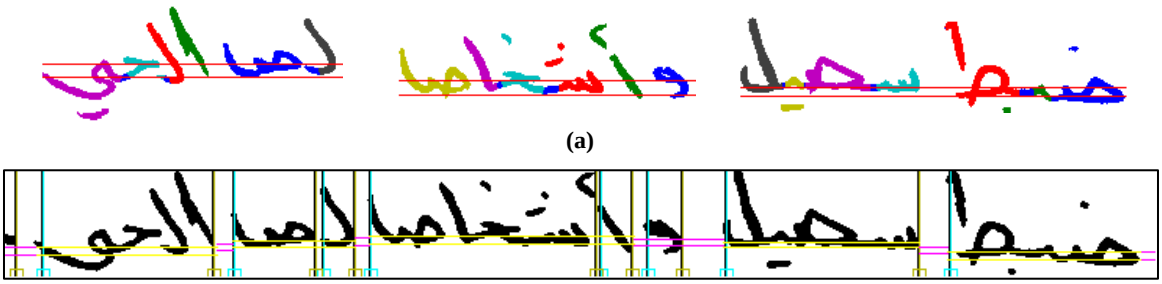
1. Let ***Segment*** be the set of chunks, where each chunk is a range of columns (c^i_1, c^i_2) that are surrounded by, but not containing, columns where **VP{c}** is less than ***cut***.
2. Let $width_i$ denote the width of the i^{th} element in ***Segment*** given by $c^i_2 - c^i_1 + 1$
3. Let ***widthWs*** be the average $width_i$
4. Assign to ***Th*** the product of ***Threshold*** * ***widthWs***
5. ***for*** the first, last and every element ***i*** in ***Segment*** where $width_i > Th$
   5.1. Compute (u_i, d_i) using **SBRE** with inputs: ***HP{c^i_1, c^i_2}, factor***
   end for
6. ***for*** every element ***i*** in ***Segment*** where $width_i \leq Th$
   6.1. Interpolate the values for the output element (u_c, d_c) from the nearest entries; ***i.e.*** $u_c = (u_{c-1} + u_{c+1})/2$, $d_c = (d_{c-1} + d_{c+1})/2$
   end for

end Algorithm **Multiple Baseline-Range Estimation (MBRE)**

Figure 31. Listing of the MBRE algorithm

BL estimation can be made more accurate if local estimations are done on chunks of lines with tuned lengths. We experimented on BL-estimation on chunks of words containing 5 or more characters taken from GTed data, like the ones in Figure 32 (a). In the absence of GT data, chunks are chosen with aid of VP. Figure 32 (b) shows vertical lines whenever the VP grows above (start preliminary chunk) or drops below (end preliminary chunk) a pre-specified threshold. Widths between starts and ends of preliminary chunks are averaged and chunks larger than a factor of the average are chosen for SBRE. BL for chunks that are not chosen in the previous step are interpolated

from BLs of their neighbors. The computed BLs are shown for different line-chunks with the lighter pairs of horizontal lines in Figure 32 (b) and the interpolated ones are shown with darker pairs.



(a)

(b)

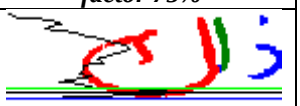
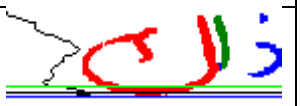
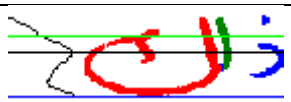
Figure 32. Chunks of words for (a) non-blind and (b) blind BL estimation.

All thresholds that were introduced in the algorithms of this section, along with the range of values that they may take, are reported in Table 21. The impact of *m* and *factor* for different values are pictorially displayed in Table 22 along with the upper and lower baseline borders and the peak of HP. We notice that *factor* affects the thickness of the range of the baseline.

Table 21. Threshold description and values used in baseline estimation.

Threshold	Algorithm	Description
m	**VP, HP**	Smoothing factor for VP and HP
cut	**MBRE**	Maximum number of pixels considered as a white cut in VP
threshold	**MBRE**	Factor to adjust the threshold on computable vs. interpolation text-line segments when compared with the average width of all segments
factor	**MBRE, SBRE**	Percentage of the maximum of HP not breaking the baseline zone

Table 22. A word with HP and upper and lower baseline borders for different *m* and *factor* values.

m	*factor 50%*	*factor 75%*
1		
5		
9		
90		

4.2 Ground-Truthing and Analysis on the Pixel-Level

The scanned paragraph images are semi-automatically ground-truthed in two levels: the text-line level, discussed in Section 4.2.1, and the character level, discussed in Section 4.2.2. Words and PAWs can be obtained by the automatic reassembly of ground-truthed characters based on the underlying text, as discussed in Section 4.2.3.

4.2.1 Line-Level Ground-Truthing

Line-level ground-truthing is performed by means of the semi-automatic tool with the interface shown in Figure 33. Native Arab users were asked to set/edit vertex points for a polygon that surrounds only the components of the topmost text-line displayed in the interface. Upon closing the polygon, all surrounded pixels are cropped to a separate text-line image and the user is allowed to request the next samples.

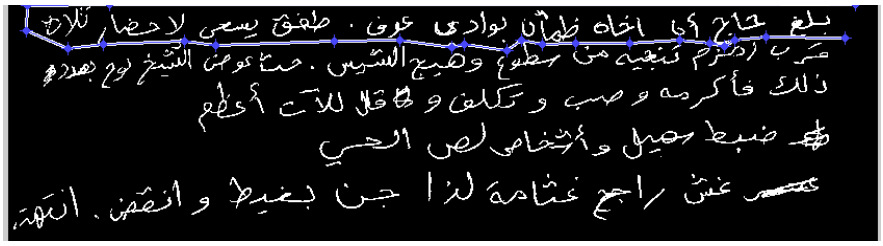


Figure 33. A snapshot of the text-line ground-truthing tool with some control points shown.

4.2.2 Character-Level Ground-Truthing

The GUI tool, shown in Figure 34, aims at character-level ground-truthing. The tool sequentially displays segments of text-lines, as defined by the VP in the MBRE algorithm, and allows the user to surround characters and ligatures with polygons. Upon closing a polygon, the enclosed pixels are labeled with a sequential number. Users are expected to choose characters in strict right-to-left order. The tool also shows the complete text-line on top of the display area to help removing ambiguities. In addition, it allows merging segments if the user demands it.

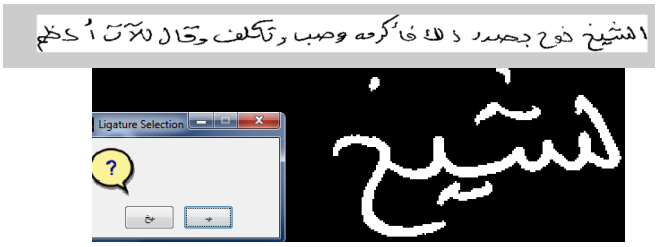


Figure 34. A snapshot of the character ground-truthing tool with confirmation request on a ligature.

Foreground pixels are labeled in the output image based on the order of their selection. Foreground pixels that are left out the polygons are labeled as Kashida. Ligatures need special treatment: By knowing the positions of the possible omni-ligatives in the corresponding text and by keeping track of the number of labeled characters per

paragraph image instance, the tool requests the user to indicate whether some polygons corresponds to a single character or to a ligature of two characters, as shown in Figure 34.

4.2.3 Words, Pieces of Words, and Extended Character-Shapes Reassembly

Most segmentation systems report their results on isolated words assuming they are somehow obtained from the dataset. To be comparable with such systems, we develop a word-reassembly tool that copies the characters of words together into separate images. Word images have an advantage in limiting error-propagation. However, they can be negatively affected by their short widths when it comes to baseline-range estimation and localized deskew. Figure 35 shows with color contrasts a GTed and re-assembled word with Kashida.

Figure 35. A GTed word with contrasting colors representing different labels.

GTed data is expensive and scarce. A total of 103 BoHs of UT PoD forms were GTed. Manual and automatic inspection filtered out mistakenly written or GTed data to remain with 54 acceptable BoHs for this work. From the discarded BoHs, 17 at least can be repaired with reasonable programming intervention. Appendix A shows statistics that can be extracted from GTed data.

Extended character-shapes are automatically assembled from labeled character-shapes by taking Kashidas which are touching to them. Figure 36 shows an example for reassembling a word and its corresponding extended character-shapes.

(a)

(b)

Figure 36. (a) A labeled word and (d) its corresponding extended character-shapes.

Ground-truthing needs human intervention, and hence is not fully automatic. To automate writing units labeling, different segmentation algorithms are discussed and evaluated in the following sections.

4.3 Blind Line Segmentation

Arabic is written in horizontal text-lines that are stacked downwards. A line either ends by a semantic stop or by reaching near the left border of the page; hence, text-lines may vary in length. Line segmentation aims at grouping pixels that belong to a line together. Line segmentation is important mainly because errors in it propagate to subsequent steps. This section presents and discusses a line segmentation algorithm for Arabic.

4.3.1 Line Segmentation Algorithm

In this section, we present an Adaptive Line Segmentation Algorithm for Arabic (ALSA) [129]. ALSA, listed in Figure 37, is used to obtain lines from BoHs of the UT PoD.

Algorithm: ALSA

Input:
- Binary image ***B*** of a BoH
- Integer ***m***, smoothing factor

Output: Labeled image **LB**, where each line of ***B*** ideally has a distinct label

1. Compute $\mathbf{HP^m\{r\}}$ from ***B***
2. Compute ***LTh*** as the average of all local minima in $\mathbf{HP^m\{r\}}$
3. Define **Valleys** as maximal chunks of rows with $\mathbf{HP^m\{r\}} \leq$ ***LTh***
4. Let **Wavg** be the average valley width
5. ***for*** each valley, **v**
5.1. ***if*** the width of **v** is less than 0.5* **Wavg**
5.1.1. merge **v** to its nearest neighboring valley
end if
end for
6. ***for*** each valley, **v**
6.1. Let $\boldsymbol{CP\{v\}}$ be the median of the longest run with minimum $\mathbf{HP^m\{r\}}$ in **v**
end for
7. Assign label **v** to pixels in **LB** corresponding to the CCs that have COGs between $\boldsymbol{CP\{v\}}$ and $\boldsymbol{CP\{v+1\}}$

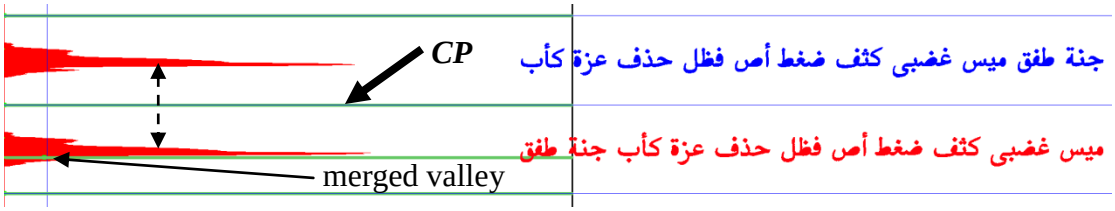
end Algorithm ALSA

Figure 37. An Adaptive Line Segmentation Algorithm for Arabic (ALSA).

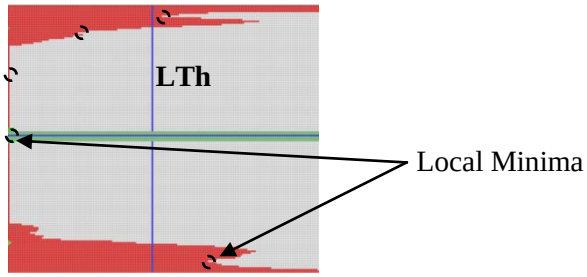
A local minimum on $HP^m\{r\}$ refers to a row with less or equal number of foreground pixels than both of its neighbors. The average of local minima, *LTh*, is used as a heuristic threshold that defines what a valley is. A valley is a maximal chunk of $HP^m\{r\}$ where values are less than or equal to *LTh*. Valleys narrower than half of the average valley width are merged with their nearest neighboring valleys.

The usage of *LTh* instead of a fixed threshold secures not dropping below the global minimum of any horizontal projection. Its disadvantage is that it is affected by the fluctuations of $HP^m\{r\}$, not only on its values. This disadvantage can be reduced by using larger smoothing factors. Within a valley, the row with minimum $HP^m\{r\}$ is declared as a

cut point *CP*. In case of a tie, the center of the longest run of contiguous CPs is taken as the *CP* of the valley, as in Figure 38.



(a)



(b)

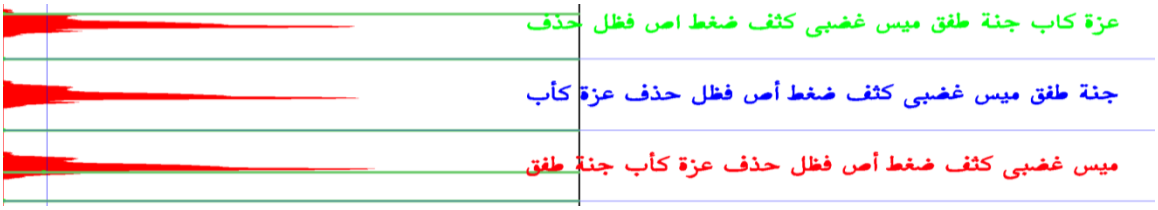
Figure 38. (a) *CP* and merged valley and (b) local minima in HP1 and *LTh*.

Finally, each connected component in the input image is mapped to a line based on the y-coordinate of its center of gravity (COG). If the coordinates fall between two CPs, *i.e.* in a valley, the corresponding connected component is assigned a distinct label of that valley. This approach avoids cutting connected components among lines.

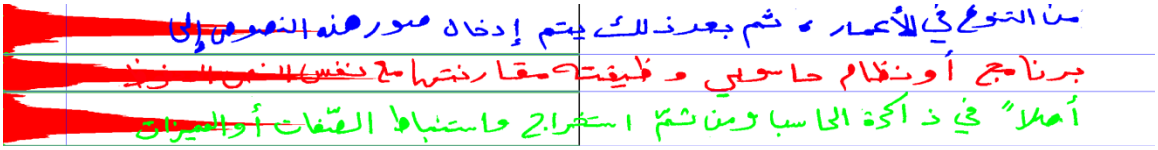
4.3.2 Line Segmentation Evaluation

Line segmentation was subjectively evaluated on 100 document images distributed among three categories: printed (provided by [134]), modern handwritten (from [135], [136]), and historical manuscripts from [137]. We took 20 images from each of the following groups: Printed *Naskh* font, printed *Akhbar* font, printed *Thuluth* font, modern handwriting, and historical manuscripts.

Figure 39 shows some results of ALSA on printed, handwritten and historical scripts. The results are displayed in Figure 39 such that components that belong to a line take a color from the repetitive set red, blue and green. **HP**[1] is displayed to the left of each output image.



(a)



(b)

(c)

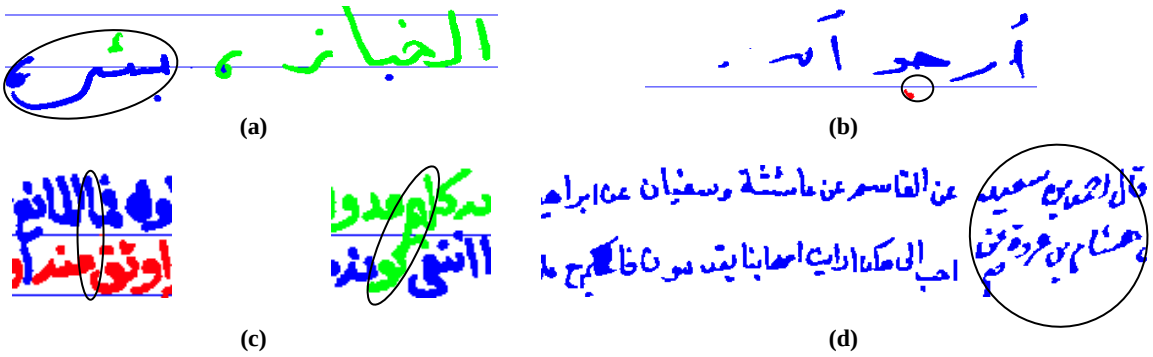
Figure 39. Output samples of (a) printed, (b) handwritten and (c) historical manuscript ALSA algorithm.

Output images are subjectively ranked on a scale from 1 to 5, where 1 refers to results without errors and 5 to severely merged or divided lines. The rankings, along with the expected value of the ranking of a category of input pages, are shown in Table 23.

Table 23. Subjective ranks of the output images of ALSA along with the expected rank per input category.

Rank	1	2	3	4	5	Average Rank
Printed Naskh font	20	-	-	-	-	1
Printed Akhbar font	20	-	-	-	-	1
Printed Thuluth font	19	1	-	-	-	1.05
Modern Handwriting	12	7	1	-	-	1.45
Historical Manuscripts	5	10	2	1	2	2.25

Most errors consist of dots and diacritics miss-line-classification (as in Figure 40(a) and (b)). Line segmentation mainly fails with skew, whether caused while writing or while scanning. Inter-line touching components (exemplified in Figure 40 (c)) are quite abundant in manuscripts. Similarly, margin writing, as encircled in Figure 40, may cause confusion to ALSA. Short lines may constitute a source of errors if not identified, or if not containing enough ascender and descender components to cover their range properly, as in the example of Figure 40 (b).



(a) **(b)** **(c)** **(d)**

Figure 40. Common error sources: (a) skew, (b) short lines, (c) touching components and (d) margin writing.

Figure 41 shows some ALSA outputs on the UT PoD of one form. Errors in line character-shape and blind segmentation are avoided if a word-dataset is used instead of a paragraph.

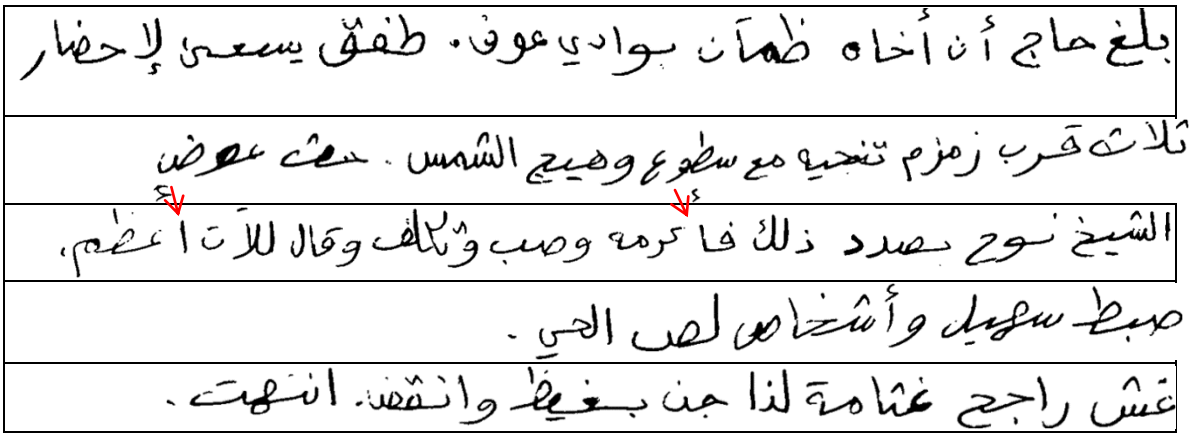


Figure 41. Output samples from ALSA on the UT part for one writer. Two arrows point erroneously assigned components to their original lines.

4.3.3 Blind Character-Shape Segmentation

Lines need to be segmented into character-shapes. We believe a line can be too lengthy to be restricted to match a GT none-blindly. Hence, we present a blind character-shape segmentation algorithm for line-to-characters segmentation.

Blind Character-Shape Segmentation Algorithm

The Blind Character-Shape segmentation algorithm dissects an image based on *valleys* in the VP, in an analogous way to how ALSA dissects paragraphs into lines, but in the vertical direction and with a few other differences. For example, the definition of a valley in Blind Character-Shape segmentation depends on two thresholds: *MountTh* and *ValleyTh.* A valley is a maximal chunk with VP values less than *MountTh* bordered by values that are also less than *ValleyTh*. Figure 42 illustrates the concept graphically.

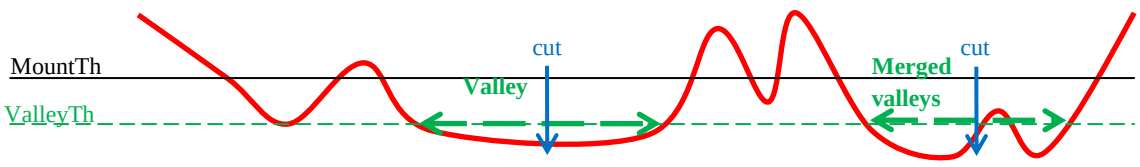


Figure 42. Sketch depicting the concept of "Valleys".

The use of two thresholds for the definition of a *valley* reduces the possibility of fluctuations near the threshold level. This idea is borrowed from Schmitt electronic switches [138]. The two thresholds can reduce turbulences (hysteresis) based on their amplitudes while the smoothing factor reduces turbulences based on their frequencies to obtain blind *word* segmentation. Figure 43 presents Blind Character-Shape segmentation algorithmically. The algorithm is described below:

Algorithm: Blind Character-Shape Segmentation

Inputs:

- Binary image ***img*** of a handwritten line or a chunk of it
- Integer ***m***, smoothing factor
- Integers ***MountTh*** and ***ValleyTh***, such that ***MountTh*** ≥ ***ValleyTh***

Output: Labeled image ***Limg***, where each character-shape of ***img*** ideally has a distinct label

1. Compute $\mathbf{VP^m\{c\}}$ from ***img***
2. Define a **Valleys** as maximal chunks of ***img*** where:
   - 2.1. $\mathbf{VP^m\{c\}}$ is never above ***MountTh***
   - 2.2. $\mathbf{VP^m\{c\}}$ at both of its ends are less than ***ValleyTh***
   - 2.3. No column **c** out of the **Valley** with $\mathbf{VP^m\{c\}} <$ ***ValleyTh*** is nearer than any other column with $\mathbf{VP^m\{c\}} \geq$ ***MountTh***
3. Define **cuts** at the median **c** from those sharing the value $\mathbf{min(VP^m\{c\})}$
4. Assign one distinct label in ***Limg*** to every pixel corresponding to the foreground of ***img*** and falling between two consecutive **cuts**

end Algorithm **Blind character-shape Segmentation**

Figure 43. Blind Character-Shape segmentation algorithm.

Step 3 defines a single cut between each two segmented character-shapes. Alternatively, the starts and ends of valleys can be used in a way similar to that of chunking words, as was shown in Figure 32(b). Single cuts are adequate to the Extended Glyphs Concatenation model while valley starts and ends are more adequate to the Synthetic Kashida Concatenation model of Section 3.2.

Smoothing, with factor ***m*** as defined in Table 21, reduces turbulences based on their frequency while Schmitt triggers reduce them based on their amplitudes. *MountTh* and *ValleyTh* are made dependent on stroke-widths. Stroke-width is computed as the

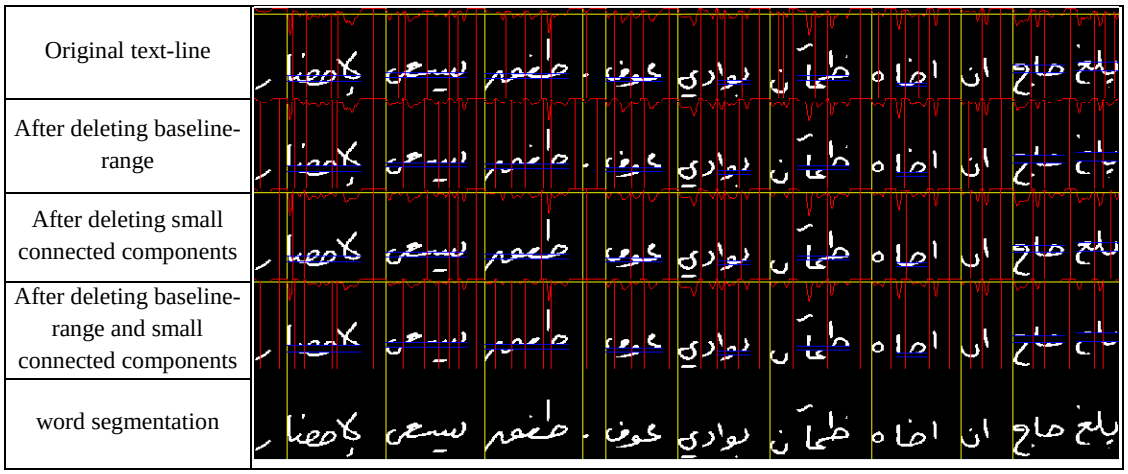
average BL ranges per BL-computed column. We multiply stroke-width by two fixed factors to get *MountTh* and *ValleyTh*. Although we still need to have two fixed thresholds, each writer can have a pair of *MountTh* and *ValleyTh* thresholds which are adaptive to his stroke width.

Character-Shape Blind Segmentation Evaluation

We have applied blind character segmentation to the original, as well as four variations of, the dataset images. The alterations include: baseline-range and/or dots removal and increased smoothing. Baseline-range removal deletes all pixels within the baseline-range, as assigned by the SBRE or the MBRE algorithms. One rationale behind baseline-range removal is that it enhances differentiating single-stroke characters from Kashida by emphasizing the role of their positions with respect to baseline region. For example, the "ر" character consists of a descending stroke that resembles a Kashida in the VP. However, if the baseline-range is deleted, such descender would remain unlike typical Kashida that would be removed.

The VP of similar characters may vary because of the different positions of dots on them. This makes it difficult to calibrate the algorithm thresholds. Dots may cause over-segmentation when they result in VP crossing *MountTh* within a character. They may cause under-segmentation if they prevent a Kashida from going below the valley threshold. Hence, removing dots, and other small connected components, becomes beneficial. We remove connected components based on area, width, and height thresholds.

A sample of the results along with the ground-truthed version is shown in Figure 44. Segmentation points for characters are shown with limited vertical text-lines for each altered version of the text. Word segmentation points are shown with text-lines that prolong from the top till the bottom of the image.



(a)

(b)

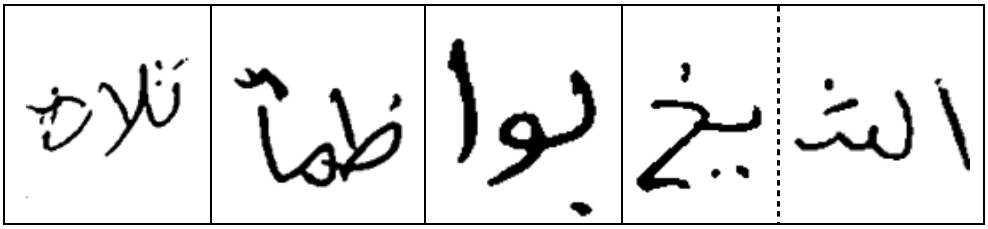
Figure 44. Visualization of (a) the segmentation results on a handwritten text-line and (b) its ground-truth.

The range of possible results from character-shape Blind Segmentation depends on: *MountTh*, *ValleyTh*, the four BL estimation thresholds displayed in Table 21, and the deletions made to input line. No exhaustive optimizations were made to these thresholds in this work. Quantitative evaluation of the results is shown in Section 4.5.

Blind segmentation for Arabic handwriting is known to have issues. The projection approach, for example, assumes that white spaces between words are generally wider than those between pieces of Arabic words (PAWs). Unfortunately, this assumption may not always hold for handwriting. Moreover, white spaces may not show in vertical projection because of inter-PAW overlap.

Secondary components and pepper noise may also bridge the vertical projection of the otherwise white cut regions. On the other hand, it is common to find broken PAWs due to salt noise. Hence; we find projection-based word and PAW segmentation impractical for handwriting. Figure 45 shows results from running ALSA on flipped lines.

An alternative approach that avoids vertical overlap obscuration aims at grouping connected component into PAWs. Main glyphs need to be recognized as PAW glyphs and all corresponding secondary and broken components need to be associated to them. Main glyphs are recognized by three features: position, size and the aspect ratio (height / width).



(a) **(b)** **(c)** **(d)**

Figure 45. Sample results from a ALSA run on flipped lines for (a) a word with two PAWs (b) a PAW, (c) a proper subset of PAWs, and (d) a word with one of its PAWs cut by salt noise.

The position of a main glyph normally touches the range of the BL. Sizes of PAWs are generally expected to be larger than sizes of secondary components. Small PAWs, like single characters, may be smaller in size than some secondary components. In particular, the character "ا" and its Hamza-versions are small characters that tend to be displaced out of (above or below) the BL range. Fortunately, most secondary components can be distinguished from "ا" by the aspect ratio feature and size, probably except for the broken vertical stroke over "ط" and "ظ".

4.4 Non-Blind Segmentation

Non-Blind techniques are those that use information of the underlying text to segment images of handwriting, accordingly. In this section, we segment words in two steps to cope with overlapping PAWs: Word-to-PAWs and PAW-to-Characters.

4.4.1 Words-to-Pieces of Arabic Words Non-Blind Segmentation

Words-to-Pieces of Arabic Words (Word-to-PAWs) segmentation receives word images and their corresponding text and aims at labeling each PAW distinctly. PAW segmentation is important because the overlap between PAWs causes errors in algorithms that intend to dissect words into character-shapes with vertical cuts. In this section, we present and discuss a Word-to-PAWs segmentation algorithm.

In Words-to-PAWs segmentation, PAWs should include their corresponding dots and diacritics. Therefore, images are first over-segmented into CCs, and then, CCs are regrouped into PAWs. Later, we use the underlying text to compute the correct number of PAWs, referred to as *correctPAW*. The algorithm listed in Figure 46 intends to identify a number of PAWs equal to *correctPAW* by classifying and reclassifying connected components into two sets for PAWs, *PAWset*, and for secondaries, *SEC*.

Algorithm: Word-to-PAWs Segmentation

Inputs:

- Word Image ***word_img***
- Word Text ***ground-truth*** corresponding to ***word_img***
- baseline parameters (***factor*** and ***m***)
- Three Thresholds:
  - Integer ***thS,*** initial minimum size of PAW
  - Integer ***thO,*** maximum orientation of a secondary
  - A structural element ***SE***, for dilatation
  - baseline thresholds ***(m, factor)***

Output: Labeled Image ***Limg*** where connected components of a PAW share a distinct label

Steps:

1. Initiate ***wmg*** to be equal to ***word_img***
2. Use ***ground-truth*** and script rules to obtain **correctPAW**, the real number of PAWs in ***wmg***
3. Let **connected components** be the set of connected components of ***wmg***
4. Let (**u, d**) be the output of SBRE for ***wmg*** with baseline parameters (***factor*** and ***m***)
5. Let *PAWset* be the set of ***connected components*** that have any point with the y-coordinate in the range (**u, d**) AND are larger than threshold ***thS***
6. Let **Secondaries** be the set of **connected components** that:
   do NOT intersect any point in the Range (**u, d**) OR
   are NOT larger than threshold ***thS***
7. Let **numPAWs** be equal to the number of elements in **PAWs**
8. ***if*** (**numPAWs** < **correctPAW** AND (**u, d**) are within the range of ***wmg*** rows
9. increment the range of (**u, d**) by performing ***u = u*** - 1 and ***d = d + 1***
10. **Repeat from** *Step 5*

else

Repeat *while* (numPAWs < **correctPAW** AND **ar *<1.5* thS*** AND **or** < ***thO*)**

11. Let **cc** be the largest element in **Secondaries**
12. Let **ar** be the area of **cc**
13. Let **or** be the orientation of **cc**
14. ***if*** (**ar <** threshold 1.5***thS*** AND **|or| < *thO***)
15. move **cc** from **SEC** to **PAWset**

end if

end while

end if

16. ***if*** (**numPAWs** > **correctPAW** AND **se** < ***SE***)
17. dilate ***wmg*** with the structural element **se**, increment **se and *goto*** *Step 5*

endif

18. Label each element in ***word_img*** AND **PAWset** distinctly
19. Use closing to label each element in ***word_img*** AND in **SEC** like its nearest neighbor from *PAWset*

end Algorithm Word-to-PAWs Segmentation

Figure 46. Word-to-PAWs segmentation algorithm.

Connected components with areas smaller than a threshold "*thS*", and far from the baseline-range are initially classified into *SEC*. Others are classified as PAW. The number of elements classified into *PAWset* can be initially different from *correctPAW*. It can become larger due to broken characters or because of secondary components being

misclassified as PAW. It can be smaller than *correctPAW* in case of touching characters or baseline-range errors.

If the number of elements in *PAWset* is less than *correctPAW*, the baseline-range is gradually expanded till a predefined limit is reached. If the number of elements in *PAWset* remains below *correctPAW*, even with expansion of baseline-range, then the baseline condition is relaxed under some additional restrictions on size and orientation values where orientation refers to the angle between the x-axis and the major axis of the ellipse that has the same second-moments as the region. It is noticed that small PAWs tend to have more vertical components than similar sized secondaries.

To overcome the effects of broken PAWs, image dilatation is performed, with several structural element dimensions, if necessary. We use a rectangular structural element that favors merging objects vertically rather than horizontally. Iteratively, we dilate all connected components. A PAW element can be a group of connected components combined when dilated. One drawback of this method is that it can sometimes merge several objects in one iteration; leading the number of elements in *PAWset* to exceed *correctPAW*. Figure 47 shows some examples of broken and corrected PAWs.

Figure 47. Examples of broken PAWs that are corrected.

Table 24 lists the thresholds used in the Word-to-PAWs segmentation algorithm and their chosen values.

Table 24. Thresholds of the Word-to-PAWs segmentation algorithm along with their used values.

Description	Values
Maximum area of a secondary connected component, ***thS***	65
Maximum orientation of a secondary, ***thO***	40^{o}
Structural Element ***shape***, and dimensions ***(rows, columns)***	Rectangular (30,16)
Baseline thresholds ***(m, factor)***	(9, 0.7)

Table 25 displays a breakdown of these errors as found in our 2,322 output words. A mistaken result might be reported in more than one category. Touching glyphs are mainly caused by writers using non-standard styles of writing. Most errors (138 errors) were of ground-truthing. Some errors were of the writing or scanning processes. Dots and broken-glyph assignments can be reduced, but not completely eliminated, by the thresholds in the algorithm.

Table 25. Words containing errors with frequency counts per error cause.

Error / Word	Touching glyphs	Secondary assignment	Broken PAW	Total
اخاه		1		**1**
الحي	2	3		**5**
الشمس	3		2	**5**
الشيخ	2		3	**5**
انتهت		3	2	**5**
أعظم		1	1	**2**
أن		2		**2**
بوادي	2	1		**3**
ثلاث		1		**1**
حاج		2		**2**
ذلك		3		**3**
راجح	1	2	4	**7**
زمزم	2	17		**19**
سطوع	3	1		**4**
ظمآن		1		**1**
عوض	2			**2**
عوف	1			**1**
فأكرمه	1	9		**10**
قرب	1	5		**6**
لذا		5		**5**
للآت		5		**5**
نوح	1			**1**
وانقض		7		**7**
وأشخاص		1	3	**4**
وتكلف		2		**2**
وصب	1			**1**
وقال	3			**3**
وهيج	1	1		**2**
Total	**26**	**73**	**15**	**114**

4.4.2 Pieces of Arabic Words to Character-Shapes Segmentation

Algorithms that segment Pieces of Arabic Words (PAWs) to characters are presented here. PAW-to-Characters segmentation takes some features from our character blind segmentation algorithm and others from our Word-to-PAWs segmentation algorithm. One PAW-to-Characters segmentation algorithm, the Fuzzy Parameters algorithm, listed in Figure 48, uses statistics on character widths (presented in Appendix A) to integrate fuzzy segmentation.

Algorithm: Fuzzy Parameters

Inputs:

- Integer ***W*** the width of image **img**
- Array ***TXT{}*** of the codes of the underlying text for **img**
- Number $\alpha \geq 1$, an attenuation factor for fuzzy range
- Array ***Ws{}*** of character width averages indexed by TXT
- Array ***STDSs{}*** of character width standard deviations

Output:

- Array **DL{}** of fuzzy centers
- Array **DR{}** of fuzzy ranges

Steps:

1. Let **sum** be the sum of all ***Ws{TXT}*** elements
2. Initiate **accumulator** to ***0***
3. ***for*** **i** indexing all characters in ***TXT***
4. **accumulator = accumulator +** $\frac{W}{sum} Ws[TXT_i]$
5. **DL{i) =** $\lfloor W -$ **accumulator** $\rfloor$ /* where **DL** is the array of fuzzy centers and $\lfloor\rfloor$ denotes the *rounding* operator */
   end for /* subtraction is to reverse the center position, as Arabic is written from right to left */
6. ***for*** **i** indexing until the penultimate character in ***TXT***
   R{i} = $\lfloor$***(STDSs{ TXT{i} } + STDSs{ TXT{i+1} }) / 2***$\rfloor$
   nR{i} = $\lfloor \frac{W}{sum}$ ***STDSs{ TXT{i} } / α***$\rfloor$ // normalize and attenuate
   end for
7. Replace **nR** values that are lower than 1 by 1 and all values that are greater than ***W*** by ***W***
8. ***for*** **i** indexing until the penultimate character in ***TXT***
9. **DR{i} = DR{i}** representing the vertices of the triangle (**DL{i}, R{i}**)), **(DL{i} - nR{i}, 0)** and **(DL{i} + nR{i+1}, 0)**
   end for

end Algorithm **Fuzzy Parameters**

Figure 48. Fuzzy Parameters algorithm for the estimation of non-blind character segmentation ranges.

Fuzzy Parameters aims at making some points more likely to have cuts solely based on the priori mean and standard deviation statistics of the character-shapes

involved in the text. The means help distributing the centers of the cut-points so that their relative positions match the relative values of the respective means. Moreover, the likelihood of a cut at a given position is inversely proportional to the standard deviations of the two neighboring widths. Around each center of cut-point, the likelihood of a cut decreases linearly as a function of the standard deviation of the width of the character to its side. The fuzzy ranges vary proportionally to the standard deviations so that if a character can have a wide range of values, we give it a larger fuzzy range. Figure 49 shows an example of the fuzzy ranges between two character-shapes when the attenuation factor, α, equals four.

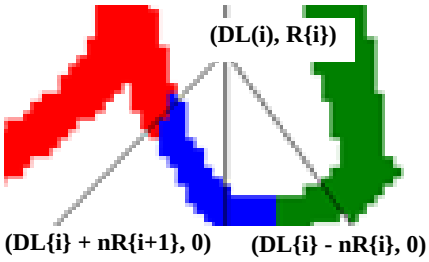


Figure 49. Fuzzification of the likelihood of cut-points between two connected characters.

The fuzzification with the parameters estimated above contributes in the PAW-to-Characters segmentation algorithm listed in Figure 50; however, only Figure 51 tells its complete story. First, we optimize the selection of mountain and valley thresholds according to the real number of characters obtained from text information. In case of ties, we use the corresponding character width means to choose more likely cut centers. In all cases, we subtract the fuzzy triangles from the VP to encourage, without forcing, the algorithm to find potential valleys near them. The disadvantage is that the subtraction may also affect mountains, which are as necessary as valleys in defining cuts. The subtraction can be constrained to affect only certain values of the VP.

Algorithm: PAW-to-Characters Segmentation

Inputs:

- Image ***PAW_img***, of the PAW to be segmented
- Integer ***m***, the averaging factor
- Text ***ground-truth*** corresponding to the characters in ***PAW_img***
- Array ***Ws{}***containing the average statistic widths of characters
- Arrays ***STDs{}*** containing the standard deviation of the widths of characters

Output: Labeled Images **Lcharacter** and **Fcharacter** where every character should have a distinct label

Steps:

1. Use ***ground-truth***, to obtain **realChs**, the ideal number of characters,
2. Use Algorithm **Fuzzy Parameters** to obtain **DL{}**, the dissection locations, and ***DR{}***, the dissection ranges
3. Let ***v*** be equal to $\mathbf{VP^m\{c\}}$ of ***PAW_img***
4. Initiate **error** to infinity
5. ***for*** integer **MountTh** ranging from ***min(v)*** ***to*** ***max(v)***
6. ***for*** integer ***ValleyTh*** ranging from ***min(v)*** ***to*** **MountTh**
7. Use Algorithm **Blind character segmentation** to obtain **cuts** on ***PAW_img***
8. ***if*** *the* number of elements in **cuts** equals **realChs**
9. update **error** with sum(|**DR- cuts**|) if it is less than the previous error

end if

end for

end for

10. Assign to pixels in each **Lcharacter** range between neighboring cuts corresponding to foreground pixels of ***PAW_img*** a distinct label
11. ***v = v - DR***
12. Repeat the Steps 4 and 5 one more time
13. Assign to pixels in each **Fcharacter** range between neighboring cuts corresponding to foreground pixels of ***PAW_img*** a distinct label

end Algorithm PAW-to-Characters Segmentation

Figure 50. PAW-to-Characters Segmentation Algorithm.

The dotted lines in Figure 51 are for blind segmentation. Solid lines with triangles at the bottom are for the fuzzy width-segmentation. Thick dashed lines are for the fuzzy hybrid segmentation. The curves on top of the image are upside-down plots for the VP (with more values) and VP – fuzzy (with less values).

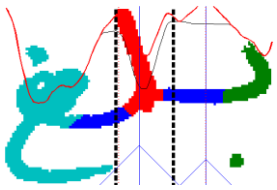


Figure 51. An example of the several character segmentation results.

PAWs-to-Character-Shapes Segmentation cannot segment ligatures. Table 26 displays potential ligatures along with the ID numbers of writers reported to use any of them. Out of 648 omni-ligatives, 73 are ligatures. Thirty out of the 54 writers never used any ligatures. Twelve writers have between one and two ligatures in their paragraphs. Ten out of the 12 potential ligatures are used. The maximum number of ligatures one writer has used is seven. We choose to leave ligatures handling for future research.

Table 26. Ligatures per writer as reported while GTing the unligative text part of our dataset.

Writer ID	ظمـ	لحـ	يخ	شمـ	يج	تهـ	شخـ	سهـ	جح	نجـ	ظم	كلـ	Total
5	1	1	1	1	1	1	1						**7**
11	1	1	1		1	1	1			1			**7**
116	1	1	1	1	1			1	1				**7**
199	1	1	1		1		1			1			**6**
9	1	1	1		1					1			**5**
85		1		1			1	1	1				**5**
119	1	1		1				1					**4**
139	1	1	1			1							**4**
201		1			1	1			1				**4**
16	1		1			1							**3**
19		1				1			1				**3**
123	1	1		1									**3**
6	1							1					**2**
7	1			1									**2**
120		1	1										**2**
2										1			**1**
3	1												**1**
88		1											**1**
105	1												**1**
106	1												**1**
109	1												**1**
140		1											**1**
185		1											**1**
187	1												**1**
Total	**16**	**15**	**8**	**6**	**6**	**6**	**4**	**4**	**4**	**4**	**0**	**0**	**73**

4.5 Segmentation Evaluation with Ground-Truth

In general, image segmentation suffers from the lack of quantitative validation methods [139]. One exception is when ground-truths are available. In this section, we

introduce an adaptation of an entropy-based image segmentation validation metric [140]. The metric cross-validates segmented images against ground-truths. We adapt the method for handwriting so that it allows any cut in the connection stroke (*e.g.* Arabic Kashida) without contributing in over-segmentation and under-segmentation errors.

First, we summarize the underlying concept. The entropy, *H(x)*, of a discrete random variable X is given by:

$$\boldsymbol{H(X) = -\sum_{X}(P(X)logP(X))} \qquad (4.3)$$

where *P(x)* is the probability of event *x* for the random variable *X*.

The conditional entropy of X given Y, *H(X|Y)*, is defined by Equation (4.4) which is equivalent to Equation (4.5)

$$\boldsymbol{H(X|Y) = -\sum_{X,Y}(P(X,Y)logP(X|Y))} \qquad (4.4)$$

$$\boldsymbol{\sum_{y\epsilon Y}(P(Y=y)logP(X|Y=y))} \qquad (4.5)$$

where *P(X,Y)* is the joint distribution of X and Y.

Let A be a segmented image and G be its corresponding ground-truth. H(A|G) is the expected entropy of the labels taken from A with pixel locations corresponding to label y in G. It detects under-segmentation errors in an image. The conditional entropy H(G|A) quantifies over-segmentation errors. Figure 52 helps explaining how we evaluate over-segmentation and under-segmentation errors from the segmentation result and the ground-truth of a word.

(a) **(b)**

Ground-truth (G)			
Corresponding Segmentation Result (A)			
H(A|G)	0	0.6469	0

(c)

Segmentation Result (A)			
Corresponding Ground-truth (G)			
H(G|A)	0	0.9831	0

(d)

Figure 52. Labeled (a) ground-truth and (b) segmentation to evaluate (c) over-segmentation and (d) under-segmentation with conditional entropy.

Over-segmentation and under-segmentation do not always imply that the resultant number of segments is larger than or lower than that in the ground-truth. Miss-segmentations resulting from the displacement of segmentation cut-points between neighboring characters, as well as those resulting from overlaps that cannot be separated with vertical *cuts*, are evaluated as over-segmentation in one character and under-segmentation in the other one. Any miss-segmentation can be quantified as a combination of over-segmentation and under-segmentation.

We adapt the metric so that the background and the connection pixels (*i.e.* white areas as well as Kashida zones, as shown in Figure 53) do not contribute to the error values. This is performed by giving Kashida of the GT the labels of their nearest connected neighbors.

Colored Segmentation Result **Colored GT (with Kashida in blue)**			
***H(A|G = y_i)* (OS)**	**0**	**0.1557**	**0**
***H(G|A= y_i)* (US)**	**0.0798**	**0**	**0**

Figure 53. Segmentation evaluation with Kashida labels.

The combination of errors from several samples is done via weighing each error value by the size of its component and averaging them. The combination of over-and under-segmentation error values, however, is generally not straightforward. It is worth noting, however, that for this metric the weight of under-segmentation is usually higher than that of over-segmentation because of the typical sizes of the erroneous components in each case.

We ground-truthed a total of 2,322 words (8,640 character-shapes) to the character-level and use the GT to demonstrate the process and results on ten segmentation scenarios. We divide the algorithms into:

- Blind-segmentation, that does not utilize text-information, and
- Non-blind segmentation, or text-alignment, that utilizes certain features of text-information.

Besides, we separate the input and output levels into: Text Line-to-Characters, Words-to-PAWs and PAWs-to-Character portions, as in Table 27. The "*Th*" column

contains values for a factor of the stroke-width of each input image. *Th* shows the values of *MountTh* and *ValleyTh* when stroke-width is multiplied by 1.1 and 0.9, respectively.

Table 27. Segmentation experiment details and results using the adapted conditional entropy evaluation method.

Algorithm Brief Description	**Input Details**	***m***	***Th*/ *x*=…**	**Over-Errors (bits)**	**Under-Errors (bits)**
Blind **Line-to-Characters**	Original text-line**s**	5	1	0.3555	**0.6121**
	baseline-range deleted text-line**s**	5	0.6	0.3373	0.6987
	Small connected components deleted **text-lines**	5	0.65	0.3237	0.6729
	baseline-range & small connected components deleted **text-lines**	5	0.6	0.3090	0.8709
	Original **text-lines**	7	0.75	**0.2579**	0.6937
Semi-Blind **Word-to-PAWs**	**Words** & count of PAWs	Table 24		0.0202	0.0402
Count aware only **PAW-to-Characters**	**PAW** & count of characters	5	---	0.2857	0.3173
Image-Blind Text-aware **PAW-to-Characters**	**Widths** of characters	7	---	0.2391	0.2339
Non-Blind **PAW-to-Characters** with Lcharacter outputs (see Figure 50)	**PAW** & **widths** of characters	7	---	0.2374	0.2500
Non-Blind **PAW-to-Characters** with Fcharacter outputs (see Figure 50)	**PAW** & **widths** of characters	7	$\alpha = 4$	**0.2280**	**0.2336**

Within each portion in Table 27, we display the experimental results in their decreasing order of over-segmentation. Moreover, we display the best result of a portion in bold, and the worst result of a portion is underlined. For the blind Text Line-to-Character segmentation portion, a tradeoff between over- and under-segmentation errors can be seen. This trend disappears in the non-blind portion, where the two error-values decrease with the injection of more text-information. Word-to-PAWs segmentation shows results which are better by around an order of magnitude than our segmentation algorithms that target character segmentation directly. The reported blind Text Line-to-Character segmentation experiments suffer less from over-segmentation than from under-segmentation. Non-blind PAWs-to-character segmentation result in comparable over-

segmentation and under-segmentation error rates. Only the best of the reported blind character segmentation results outperformed the worst of our blind character segmentation algorithms in over-segmentation. The one-by-one images, along with their over- and under-segmentation error values, are found in our online dataset samples [141].

CHAPTER 5

ARABIC HANDWRITING SYNTHESIS

Handwriting synthesis refers to the computer generation of online and offline data that resembles human handwriting. It aims at transforming input text into images of handwritten samples with equivalent script, whereas recognition maps handwritten samples into digital text. In this chapter, we present concatenation synthesis techniques and a statistically-modeled connection stroke. We introduce *selection-based* concatenation that selects character-shape samples according to their feature matches and some distance measure.

The approach is outlined in the block diagram of Figure 54. The approach takes character-shape images as inputs and reshuffles them into instances of any requested text. For each character-shape, features are extracted at the connection-points to ensure the selection of samples that are most compatible for connection. The filled rectangles in the diagram show the four steps of the synthesis procedure while the rounded rectangles indicate the information needed in each step.

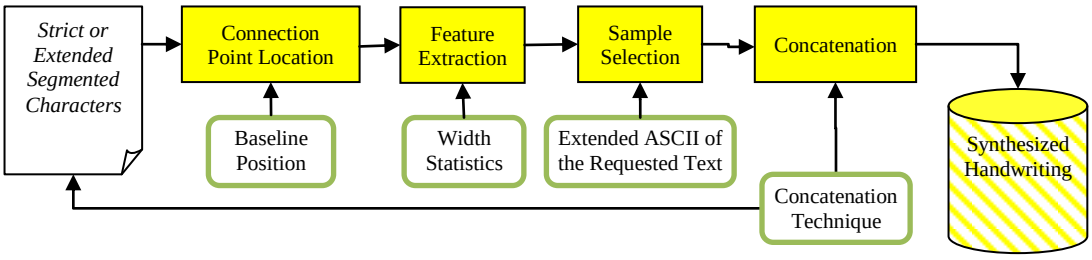


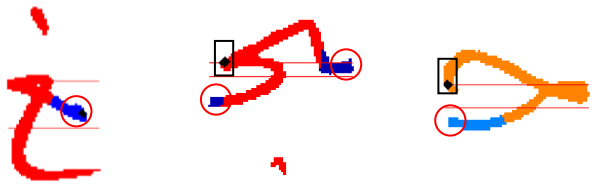
Figure 54. Block diagram of the steps to obtain an image dataset in filled boxes.

The input dataset contains character-level ground-truthed texts that cover all of the Arabic character-shapes. From the dataset, we extract strictly segmented character-shapes that minimize the extension part out of the character's glyph, as well as extended character-shapes. These are used for the Extended-Glyphs and the Synthetic-Extensions concatenation techniques. The connection-point location step intends to find the coordinates of the connection edges for each character-shape instance, sometimes with the help of the baseline information obtained before segmentation. Thickness and directions features are computed for connection parts. In addition, the sample-to-average character-shape width ratio is also used as a feature to help choosing character-shapes of similar scales. Based on their features, samples of the text to be synthesized (entered from a keyboard or a file), particular character-shape samples are selected. Finally, the selected samples are positioned on a canvas using one of two connection schemes: extended-glyphs (EG) concatenation and Synthetic Extensions (SE) concatenation.

5.1 Connection-Point Location

Connection-point location is necessary for feature extraction, sample selection, and PAW connection. We investigate connection-point location for two scenarios: the blind scenario and the ground-truth aware scenario; the former being prone to the baseline zone (BL) errors and the latter to ground-truth errors.

The extensions can be methodically located from character-shape images based on their right and left edge positions. We search for connection-points at the right side of (E) and (M) character-shapes and at the left side of (B) and (M) character-shapes. Examples of (B), (M) and (E) character-shape extensions are shown in Figure 55.



(a) (b) (c)

Figure 55. Correct extensions (in circles) and erroneous extension location samples (in rectangles) for (a) an ending character-shape, (b) a middle character-shape and (c) a beginning character-shape.

Figure 55 also shows that not only connection parts, but also character parts, may come at the edges of a character-shape image. This case causes ambiguities in connection-point location. To reduce such ambiguity, we choose points within the baseline-range (BL). BL cannot be accurately estimated from single character-shapes. Hence, we receive BL information for chunks of characters from the previous steps. We identify an extension as the nearest connected component (CC) to the bounding box edge side of interest within BL and not farther than *N* pixels from the edge itself. From that CC, the connection level is taken as the y-coordinate of the median of nearest column to the corresponding edge.

We report error rates for connection-point location based on 1,462 character-shape images that have the Kashida label near their right and left sides. The right and left error rates of our approach are 1.64% and 2.12%, respectively. Some errors are due to inaccurate BL estimations, and some are due to ligatures, a case in which characters connect out of BL.

5.2 Feature Extraction

We extract features that describe the connection-parts (Kashida features) and the relative-widths of the character-shapes (Width feature). Kashida features are intended to

assure within PAW matching. They measure the thickness and the direction of connection-parts within a window of *N* pixel-columns from the outer edge of a character-shape. The thickness feature at Column *j* is taken as the vertical distance between the upper and the lower contours of the connection-part. The direction feature is taken as the difference between the middle y-coordinate of the connection part pixels at Column *j* and the corresponding value for Column *j+1*. Hence, *N* thickness features and *N-1* direction features can be computed per connection-part. Kashida features are illustrated in Figure 56 (a). The Width feature refers to the ratio of the width of a character-shape sample to the average width of its class samples. The average widths per sample are pre-computed and stored for use in this feature. Figure 56 (b) illustrates the effect of the Width feature.

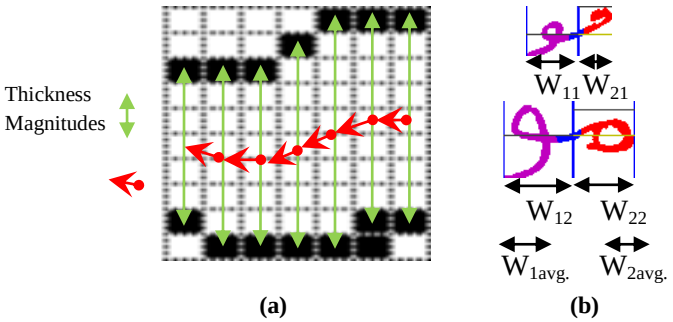


Figure 56. Illustration of (a) the features of the 7 leftmost pixels of a left connection part and (b) the two matches based on the width-ratio feature.

We compute and store all features in *2×N* sized structure. Kashida features are stored so that the outer features of the rightmost connection-part are matched with the inner features of the rightmost connection-part. The different Kashida features are stored in different structures to ease taking subsets with window sizes less than *N*, if needed. The width-ratio features are matched together, regardless of their connection-part sides.

The Width feature typically has smaller values than thickness values. To maintain significant effects for the Width feature, it is multiplied by a pre-specified

weight, *WT*, and the Kashida features are normalized by their respective numbers. Next, we select representative samples of character-shapes for synthesis.

5.3 Sample Selection

Samples of character-shapes contributing to the synthesis of some text are selected so that they collaboratively pursue a natural look and behavior. The features of neighboring samples are evaluated by the city block distance measure. The collection of samples that minimizes the sum of the measured distances is selected. When synthesizing several versions of a word, we assure each selection is unique.

The search space of sample selection is affected by the number of *units* to be jointly selected (*U*) and by the number of samples per character-shape. *Units* refer to extended-glyphs in EG concatenation and to character-shapes and SE. We estimate the number of comparisons required for a selection by *Comparisons(U)*, the number of distance matchings for a unit of *U* character-shapes. In the following, let U_i be the number of samples of the i^{th} character-shape in the synthesized unit. Equation (5.1) estimates the search space for brute-force selection.

$$\boldsymbol{Comparisons(U)} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{U}} \boldsymbol{U_i} \qquad (5.1)$$

Brute-force search for sample selection is impractical except for small values of *U*. One solution to this problem is to limit the usage of brute-force selection to PAWs, since more than 99.5% of PAWs consist of 5 or less character-shapes [101]. Then, the

different PAWs are linked based on the width features of their two neighboring characters.

Another approach that avoids intractable brute-force selection is the forward algorithm [142] that performs optimal matching for the first pair of the character-shapes and sequentially matches neighboring character-shapes in a chain Equation (5.2) represents the number of vector comparisons for our greedy forward algorithm.

$$\boldsymbol{Comparisons(U) = U_1 \times U_2 + \sum_{i=3}^{U} U_i} \qquad (5.2)$$

Curtailed and broken connection parts may result in thickness values of zero. When matching features-structures for sample selection, the zero thickness features may undesirably match. For this reason, we penalize zero-thickness extension parts by replacing their distances by larger values.

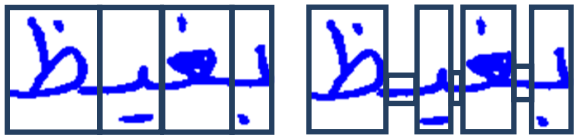
5.4 Concatenation

In this step, images of cursive text are composed from individual character-shape samples. This is accomplished through one of two concatenation approaches: the Extended Glyph approach (EG) and the Synthetic-Extension approach (SE).

The aggregation of the character-shape with part of its attached Kashida, as shown in Figure 57(a), is referred to as an extended glyph, and it is the basis of the EG approach. Extended-glyphs can be of the beginning, middle or ending shapes, denoted as (*Bx)*, *(xMx)* and *(xE)*, respectively; where the '*x*' prefix/suffix indicates the presence of a Kashida extension before/after a character-shape. The regular expression of a multi-

character PAW under this model is given by *(Bx)(xMx)*(xE)*, where the '*' mark indicates zero or more occurrences of the symbol before it.

On the other hand, SE concatenation utilizes synthetic Kashida between strict character-shapes that were extracted with minimal Kashida extensions, as shown in Figure 57(b). The regular expression for SE concatenation is given by *(B)(K(M))*K(E)*. The search space of samples can be larger in SE than in EG due to the greater number of units in SE. Below, we discuss some issues of the EG and the SE models.



(a) **(b)**

Figure 57. Examples of (a) Extended-glyphs connection model and (b) Synthetic Extensions connection model.

5.4.1 The Extended Glyph Approach

Extended-glyphs are extracted from the dataset as the character-shapes along with their neighboring Kashida extensions. Then, the Kashida extensions are trimmed so that they are only few (2-6) pixels out of the extended glyph. Trimming extensions of the extended character-shape model not only keeps the extension length natural, but also leaves the connection-point at a clean cut.

The EG model uses direct-connection concatenation to synthesize PAWs and no-connection concatenation between PAWs. Extended character-shapes are placed in juxtaposition where character-shapes within a PAW are vertically aligned so that their horizontally extensions overlap with *N* pixels.

Then, spaces are added between PAWs and words. If the text to be synthesized explicitly specifies a space, a gap size from the uniform distribution between 14 and 28 pixels is selected and a corresponding space is inserted in the synthesized image. Displacements in both the gapping and overlapping directions are made between PAWs. The displacement values are selected from a normal distribution centered after (E) and (A) character-shapes by 5 pixels and scaled by a standard deviation of 1.75. Cleary, it favors gaps over overlaps.

5.4.2 The Synthetic-Extension Approach

The Synthetic-Extension (SE) model uses a synthesized connection stroke to concatenate strictly-segmented characters into PAWs. Apart from the strict segmentation and the synthetic extension, the procedure is similar to that of EG. In this section, we explain connection stroke modeling from analysis to synthesis

A statistical model learns Kashida shapes from our dataset. It analyses the features of extracted Kashida and captures them into discrete histograms that are sometimes loosely referred to here as Probability Density Functions (PDFs). These PDFs are later used to draw values for a synthesized Kashida. The following sections elaborate on Kashida extraction, representation and modeling.

Kashida Extraction

Kashida extensions are extracted from the dataset based on their ground-truth labels. All Kashida and noise components share a common label value. Hence, to isolate Kashida from pepper noise components, we constrain the extracted components to be

adjacent to two consecutive characters. For some later statistics, the names of the neighboring characters are stored along with their corresponding Kashida.

To assure accurate Kashida analysis, the left and right borders need to be cleanly (vertically) cut. To achieve this, we trim slices from both sides Kashida borders. The widths of the slices are adaptively computed based on the Kashida width. Some Kashidas are discarded based on size and aspect ratio thresholds. Figure 58 displays samples of trimmed and discarded Kashida.

Figure 58. Samples of (a) trimmed Kashida and (b) discarded Kashida.

Kashida Representation

Each extracted Kashida is represented by three sets of features: its width (Width), the directions of its upper contour (UCD) and the directions of its lower contour (LCD). Figure 59 shows these features. Note that we do not need to model Kashida starting or ending thicknesses for our connection scenarios.

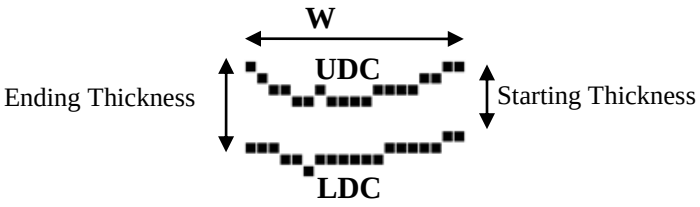


Figure 59. Kashida width (W), upper contour directions (UCD) and lower contour directions (LCD).

We show how we identify Width, UCD and LCD of our previously extracted Kashidas as in the algorithm that is listed in Figure 60. Kashidas contribute with more

than 1,000 widths and 13,000 pixel-directions (slopes) for each of the UCD and LCD. In the next section, we discuss computing the PDFs for these features that represent Arabic Kashida for different scenarios.

Algorithm: Kashida Features Extraction
Input: Arabic Handwritten Images with Character-Level Ground-Truth Labels
Output: Arrays for the values of Width, UCD and LCD
Procedure:
For each labeled image *i*:
For each Kashida connected components *K:*
Filter out noise based on dimensions & aspect ratio
Trim leftmost and rightmost pixels of *K*
Compute and Save the width *Width* of *K* in **Array Widths**
Compute $\frac{dK}{dy}$ to obtain upper and lower contours of *K*
/ UCy: vector containing the pixels' y-coordinates of upper contours of K and LCy: vector containing the pixels' y-coordinates of lower contours of K */*
Compute $\frac{dUC_y}{dx}$ and $\frac{dLC_y}{dx}$ to obtain **UCD** and **LCD**
// The pixel slope vectors of *UD* and *LD*
Reverse *UCD* and *LCD* to become from right to left
Multiply *them* by -1 to comply with coordinate system
Save UCD and LCD in **Arrays** UCDs and LCDs
END Algorithm

Figure 60. The Kashida features-extraction algorithm.

Probability Estimation

The probability density functions (PDFs) for Width, UCD, and LCD of Kashida are computed for subsets of the Kashida population, as well as for their proper set. Kashida subsets may be taken per writer, per the character they emerge from, or by the character they reach.

Two types of PDFs are estimated: Kashida Width PDFs (KW-PDFs) and Contour Direction PDFs (CD-PDFs). KW-PDFs are estimated based on bins that are eight pixels wide. Strokes shorter than 6 pixels are discarded in the extraction step; hence, the first bin is usually under-populated. CD-PDFs are estimated for the upper and the lower contours. We show upper CD-PDFs (UCD-PDFs) for the upper contour of a whole

Kashida as well as for each of five equal portions of it. We also show UCD-PDFs that are conditional on the predecessor contour-pixel direction value. Lower CD-PDFs (LCD-PDFs) are either estimated independently or conditionally given the corresponding upper contour direction.

The PDFs we present are first estimated on the complete set of Kashida, and then they are re-estimated on subsets based on the connected characters or the writers. Per-connected-characters' PDFs are presented once per the predecessor character and again per the successor character of a Kashida. The fourth set of Kashida for which the PDF is estimated is the per-writer subset.

Three main types of PDFs are estimated for all of the subsets of Kashida. In particular, KW-PDFs, 5-Portions UCD-PDFs and Conditional-on-Upper LCD-PDFs are considered. CD-PDFs that are conditional on the predecessor contour-pixel are unstable when used to synthesize Kashida because the PDFs choice is determined by a single random value. Table 28 lists all Kashida PDF types per their subsets. Together, these PDFs contribute 2,459 Width and contour values.

Three row sets can be identified in the table: the width PDF, the UCD set, and the LCD. We choose one PDF type from each of the latter two sets. The 5-portioned UCD was chosen because it is more robust than the conditions UCD, which makes the pixel direction solely conditional on one previous pixel direction. To link LCD to the corresponding UCD, we compute conditional PDFs of LCD given UCD.

Table 28. The computed PDFs and their sizes per Kashida subsets and types.

Sets / PDF	Statistic per Kashida	Proper set	Subset per previous character-shape	Subset per next character-shape	Subset per Writer
KW	1	1×1	42×1	**50×1**	44×1
UCD	W	1×1	42×1	50×1	44×1
Conditional UCD	(W-1)	1×5	42×5	50×5	44×5
5-Portioned UCD	(W/5)	1×5	42×5	**50×5**	44×5
LCD	W	1×1	42×1	50×1	44×1
Conditional LCD	W	1×5	42×5	**50×5**	44×5

Figure 65 to Figure 65 show examples of the histograms for the proper set of Kashidas.

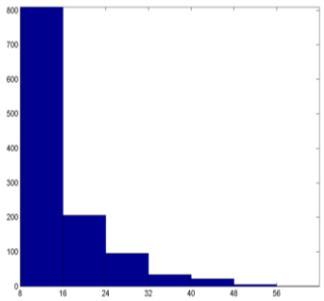


Figure 61. Kashida Width histogram for the proper set of Kashidas (KW-PDF).

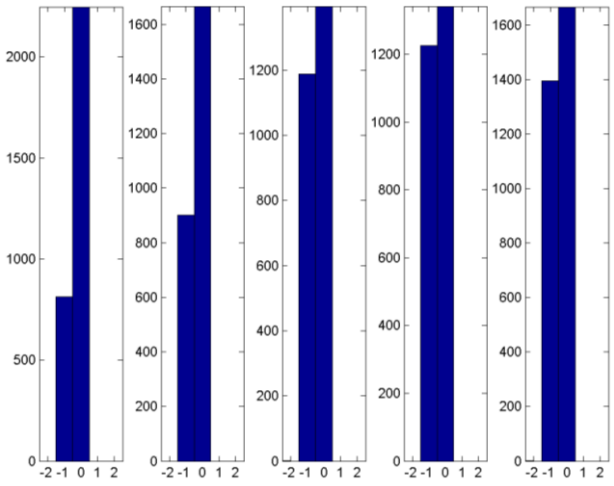


Figure 62. 5-Portioned Upper Contour histograms for the proper set of Kashidas (UCD-PDF).

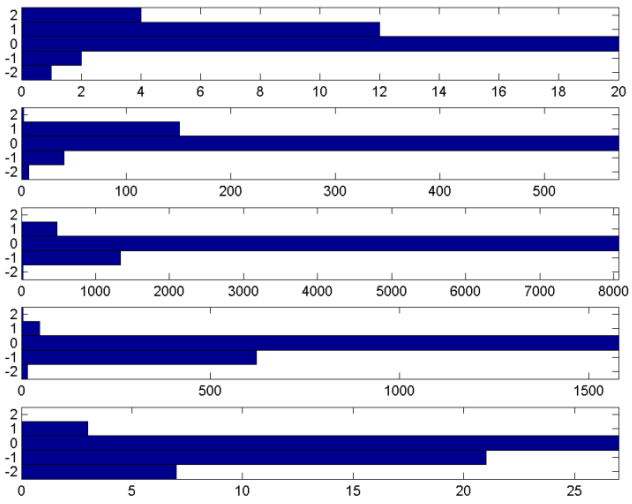


Figure 63. Conditional Lower Contour Directions histograms for the proper set of Kashidas (LCD-PDF).

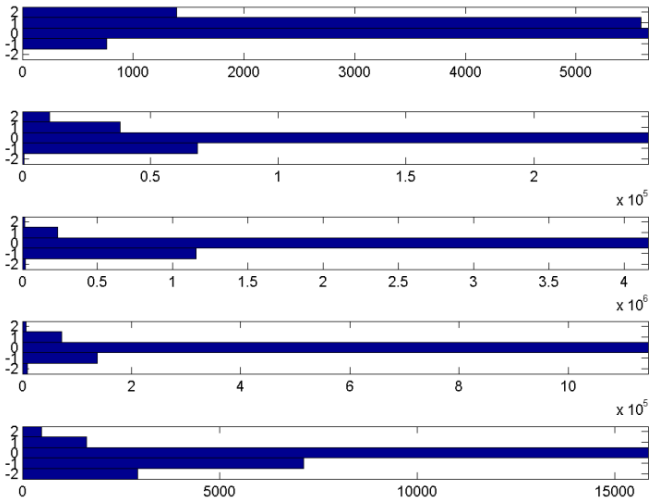


Figure 64. Conditional histograms for the proper set of Kashidas UCD-PDF, and (e) UCD-PDF.

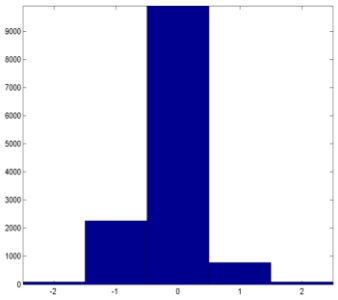
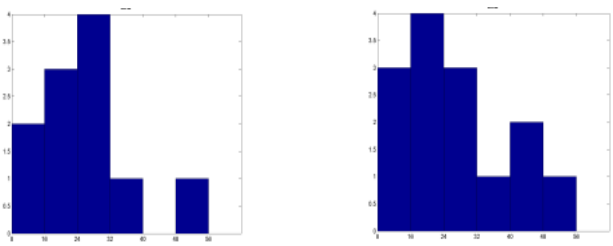


Figure 65. Upper Contour Directions histogram for the proper set of Kashidas (UCD-PDF).

Upon inspection, we noticed that the "conditional on the next character" column captured writing styles that are calligraphically justifiable. For example, the width-histograms of character-shapes ـضـ and ـظـ, shown in Figure 66, were non-descending. These two characters are often written in a special way that this subset reflects.

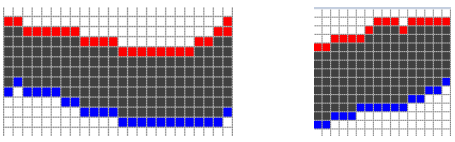


(a) **(b)**

Figure 66. Non-descending KW-PDFs found to enter to (a) Middle ـضـ and (b) Middle ـظـ character-shapes.

Kashida Synthesis

To synthesize a Kashida, we first draw a width, *W*, from the KW-PDF and add a random integer ranging from zero to the bin size to it in order to cope for the histogram quantization. Then, we draw *W* UCD values from the 5-portioned UCD and *W* other values for their corresponding values conditional-on-upper LCD and use these as the contours of the Kashida. We impose minimum and maximum distance between each UCD and its corresponding LCD values so that the Kashida thickness is always within the pre-specified range. Once the contours are selected, we fill the range between them with black pixels. Two samples are show in Figure 67.



(a) **(b)**

Figure 67. Synthesized Kashida (a) with the overall upper contour PDF and (b) with the portion-wise upper PDFs.

5.5 Experimentation and Results

Synthesis systems should be evaluated based on their intended applications, as shown in Section 2.1. Our aim in this dissertation is to improve a recognition system with natural-looking data. Hence, we present the results of our handwriting synthesis system by images and by reporting their impact on the performance of a state-of the-art text recognizer [9]. In section 5.5.2, we present the recognition results of an HMM-based system, on the popular IFN/ENIT benchmark database [108], with and without the injection of synthesized data. In this section, we present the settings and results of our synthesis system and their impact on the recognition system.

5.5.1 Synthesis Experimentation and Results

To evaluate the natural-looking of the synthesized data,we synthesize six versions of the possible multi-word names of 721 Tunisian towns/villages from our dataset described in Chapter 3. In Figure 68, we show some samples of the handwritings of our writers for convenience. Table 29 displays some statistics on the text that we synthesize.

Figure 68. Text samples of our dataset by different writers.

Table 29. General statistics on our synthesis test bed.

Feature	Value
Total PAW	1,445
Total character-shapes	3,847
Avg. number of character-shapes per town name	5.34
Maximum number of character-shapes in a town name	13
Avg. number of character-shapes per PAW	2.66
Maximum number of character-shapes in a PAW	7
Avg. number of PAWs per town name	2.00
Number of PAWs with 1 character-shape	64
Number of PAWs with 2 character-shapes	709
Number of PAWs with 3 character-shapes	422
Number of PAWs with 4 character-shapes	174
Number of PAWs with 5 character-shapes	55
Number of PAWs with 6 character-shapes	19
Number of PAWs with 7 character-shapes	2

A set of parameters affects the quality of synthesis and the time it consumes. These parameters are shown in Table 30. To synthesize unique versions of the same word, a selected character-shape combination is kept in a list and prevented from appearing again.

Table 30. Setup parameters for the synthesis.

Setting	Value
Brute force selection until (in character-shapes)	2
Zero-thickness penalty	Yes
WT weight for the W/W_{avg} features	10

Figure 69 shows some samples of the results of our extended-glyphs and synthetic-extension synthesis. With our features, we noticed that the connections of the EG images are smooth enough to fool the native eye. The images synthesized by the SE technique have more variability in shapes due to the parameters selected from the PDFs.

(a) (b)

Figure 69. Samples of our (a) extended-glyphs and (b) synthetic-extension synthesized images for three city names of IFN/ENIT.

5.5.2 Recognition Experimentation and Results

Researchers use synthesized data to expand the training set of a recognition systems and hence enhance its recognition rate [10]–[14]. In this section, we intend to demonstrate the possibility of benefitting from the injection of synthesized data into the training set of recognition systems. Our baseline system is trained on the 2,322 word samples from the dataset described in Chapter 3. We, then, assess the impact of injecting synthesized data to the baseline system. We inject samples of the EG concatenation model for one set of experiments and samples of the SE concatenation model for another

set of experiments. SE results are better than GE results due to their components' variability. Then, we evaluate our system on Set 'D' and Set 'E' of the IFN/ENIT benchmark consisting of 937 city names. Some samples from IFN/ENIT are shown in Figure 70.

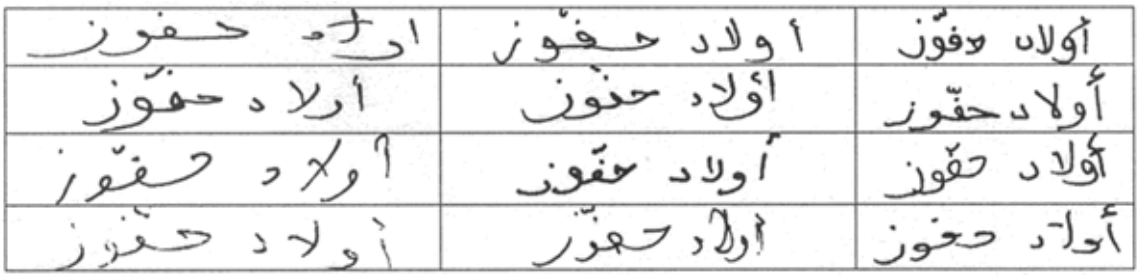


Figure 70. A town/village name written by 12 different writers.

Our text recognition system is a continuous HMM system using the HTK tools [142]. A left-to-right continuous Hidden Markov model (HMM) of Bakis topology with constant number of states per character-shape recognizer is used. We extracted nine statistical features from the word images. These features are adapted from [9] and [143]. We append nine derivative features to the original features such that the dimension of the feature vector is 18. Each character-shape HMM is modeled with the same number of states. The optimal number of states is decided based on the evaluation results.

We experiment on incremental numbers of injected data and summarize the results in Table 31. We report the top 1 word recognition rates (WRR), along with the statistical significance of the 95% confidence level, the top 5, and the top 10 best results. After six samples per city name, the change in WRR halts being statistically significant.

Table 31. Results of injecting different number of 'SE' synthesized samples in the original training data.

Number of samples injected for each of the 721 city names	*Word Recognition Rates*			
	Top 1	***Statistical significance***	***Top 5***	***Top 10***
Zero sample (Baseline System)	48.52	*(±1.00)*	64.17	67.74
One sample	64.51	*(±0.97)*	78.09	81.67
Two samples	66.76	*(±0.95)*	81.05	84.09
Three samples	67.86	*(±0.94)*	81.66	84.68
Four samples	69.00	*(±0.94)*	82.67	85.38
Five samples	69.18	*(±0.94)*	82.05	84.89
Six samples	**70.13**	***(±0.93)***	**82.94**	**85.53**
Seven samples	69.82	*(±0.93)*	82.62	85.42
Eight samples	69.29	*(±0.93)*	82.54	85.55
Nine samples	69.74	*(±0.93)*	82.89	85.59
Ten samples	70.58	*(±0.92)*	84.22	87.03

The WRR trend with number of injected images for each city name is graphically shown in Figure 71.

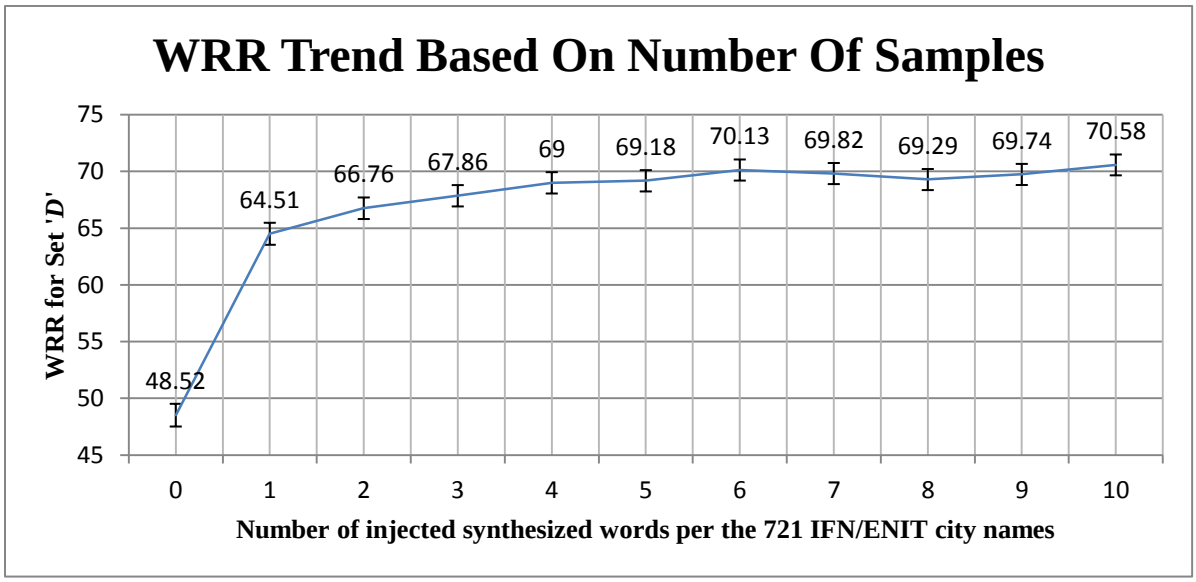


Figure 71. Recognition result and significance for injecting different number of 'SE' synthesized samples in the original training data.

Table 32 shows that the EG technique reports a WRR of 63.67%, an improvement of 9.93% whereas the SE technique reports a WRR of 70.13%, an improvement of 16.39% over the baseline system, and an improvement of 6.46% over the

EG technique when tested on Set 'D'. It shows the same trend when tested on Set 'E'. It can be clearly seen from the table that adding synthesized training data to the baseline training set significantly improves the results. Both, the EG and the SE techniques, lead to significant improvement although SE lead to a better improvement. In order to make sure that the improvements are indeed due to the synthesized data and not only due to simple addition of more data, we conducted one more set of experiments where we doubled the baseline training data by simply adding a copy of the baseline images. The results using the double number of training samples did not show any significant improvement over the baseline system; thereby further corroborating the conclusions drawn on improvements due to synthesized data.

Table 32: Word Recognition Rates (WRR) for text recognition task on IFN/ENIT database.

Training \ Testing	*Set 'D'*				*Set 'E'*			
	Top 1	***Statistical significance***	***Top 5***	***Top 10***	***Top 1***	***Statistical significance***	***Top 5***	***Top 10***
Baseline System	53.74	*(±1.06)*	64.17	67.74	48.52	*(±1.00)*	67.31	70.35
Doubled Baseline System	53.82	*(±1.06)*	64.29	67.86	48.44	*(±1.00)*	67.30	70.29
Expanded by EG synthesis	63.67	*(±1.05)*	74.44	77.98	58.54	*(±0.97)*	77.65	80.67
Expanded by SE synthesis	**70.13**	***(±1.01)***	**81.19**	**84.19**	**66.51**	***(±0.93)***	**82.94**	**85.53**

In Table 33, we show our results in the context of other comparable research. Some researchers, *e.g.* [35] [11], [38], inject synthesized data to improve the original results, whereas others, *e.g.* [10], [14], [37], experiment on the synthesized data only without the original data. Bayoudh *et al.* [35] experiment on online writer-dependent lowercase-character Radial Basis Function Network (RBFN) and Support Vector Machines (SVM) recognizers. They inject 300 synthesized versions of the 26 English

characters to the training set. Their best improvement increases the character recognition rate (CRR) by approximately 13 percents. Helmers and Bunke [10] generate data that performs approximately as well as collected data on an offline HMM recognizer whereas Varga and Bunke [11], [41] perturb handwritten text-lines to expand their training set and improve their recognition rate. Saabni and Sanaa intent to find PAW recognition rates (PAWRR) for alternative online and offline training sets [14], [37]. They evaluate their work on a Dynamic Time Wrapping (DTW) online recognizer [144] an adaptation of it for offline recognition. Similarly, Miyao and Maruyama synthesize a virtual Hiragana dataset that performs comparably to their original dataset [12]. The robustness, speed and performance of the recognition of online gestures are addressed in [91].

Table 33: Summary of our work in the context of other related work.

Citation	Task	Original DB	Original Results	Synthesized Data	Synthesized Data Results	Classifier	Injection
This Work	• Offline • Text recognition • Arabic	• 2,322 words	53.7% WRR	6 versions of 721 IFN/ENIT city names	70.1% WRR	HMM	Yes
Bayoudh *et al*. [35]	• Online • Writer-dependent recognition • Lowercase • Isolated Latin characters	• Twelve Writers • Each writing 10 versions of the 26 lowercase characters	≈ 82.5% CRR	• Generation of character 300 versions • Distortions and Analogy	≈ 95% CRR	RBFN	Yes
				• Image Distortions	≈ 92.5% CRR		
				• On−line and Image Distortions	≈ 94% CRR		
			≈ 93% CRR	• Generation of character 300 versions by: • Distortions and Analogy	≈ 96.5% CRR	SVM	Yes
				• Image Distortions	≈ 95.5 CRR		
				• On−line and Image Distortions	≈ 95.75 CRR		
Helmers and Bunke [10]	• Offline • Text recognition • Latin script	• Subset of the IAM database • Total of 1190 words • From a lexicon of 37 words • Training 80% • Evaluation 20%	70.8% WRR	• Concatenation –based synthesis of original data	65.8% WRR	HMM	No
				• Segmented character samples	68.5% WRR		
				• *n*-tuples of characters	71.1% WRR		
Varga and Bunke [11], [38]	• Offline • Writer-Independent • Cursive • Handwriting Recognition	• 3,899 word instances • 6 writers • From a lexicon of 412 words	33.1% WRR	• 5 synthesized lines added to each original line • All distortions	Substantial change 49% WRR	HMM	Yes
				• Line level geometrical transformations	47.1% WRR		
				• Connected component level geometrical transformations	38.7% WRR		
Saabni and Sanaa [37]	• Online • Arabic • PAW-recognition	• 500 PAWs • 6 different writers	81% PAWRR	• Same database synthetically generated • Using concatenation	82% PAWRR	Elastic Matching Technique	No
Saabni and Sanaa [14]	• Online and Offline • Arabic • PAW recognition	• From 48 experts • ADAB (online) 2 200 PAWs 16 356 shapes	• 80.21% precision	PAWs generated from ADAB and experts' data	In precision. • 81.16% synthetic • 81.09% user-synthetic	DWT	No
		• From 48 experts • IFN/ENIT (offline)	• 78.21% precision	PAWs generated from IFN/ENIT and experts' data	• 78.64% synthetic • 80.43% user-synthetic		
Miyao and Maruyama [12]	• Off-Line • Character Recognition • Japanese Hiragana	• 10 Japanese Hiragana characters • 50 writers • 50 real	≈ 97.5% CRR	• 50 samples • Using on-line affine transformation	≈ 97.3% CRR	SVM	No
Plamondon *et al*. [92]	• Online • Gestures • Robust increments in training	• 11 Gestures • 7 Writers • 100 each	≈ 27.5% Error rate	• 10 synthetic gestures for each real learning sample	50% Error Reduction	Evolve++	Yes

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

Handwriting synthesis has applications that target recognition systems, the human eye, or both. In this dissertation, we explore the effect of Arabic synthesized handwriting on a text recognition system. We show that the injection of segmented and re-concatenated Arabic characters significantly improved a recognition system that was otherwise trained only on the collected samples. The improvement is shown to be due to the synthesis operations rather than to the mere repetition of the same data.

Synthesizing training sets can increase the variability of character-shapes, of their connections, or of both for a given handwriting dataset. Synthesis by concatenation of Arabic characters mostly adds to the variability of the connections between character-shapes, as well as the spacing and overlapping between them. It plays a role in enhancing the robustness of explicit or implicit segmentation, independently from the underlying system. Synthesis by concatenation is particularly useful for holistic recognition systems where under-represented patterns of a certain vocabulary can be needed.

We designed a comprehensive dataset of unligative character-shapes. We collected Arabic character-shapes from their natural flow within words. Then, we developed and evaluated several character segmentation and alignment schemes to separate them. It is worth noting that our character evaluation framework can be of benefit for benchmarking the currently open problem of Arabic character-segmentation.

We synthesized handwriting from extended and strictly-segmented character-shapes. Extended character-shapes contain some connection extensions before/after the character body. They can be selected and connected directly, without need for explicit connection strokes between them. Strict character-shapes contain the character body without or with minimal extensions; hence, they need connection strokes between them. We model and generate synthetic connection strokes for this aim.

The connections stroke is modeled by estimating discrete probabilities for the following parameters: the stroke width, the upper contour direction of each of 5 equal portions of the stroke entering to a specific character-shape, and the lower contour direction conditional to the corresponding upper contour direction value. While synthesizing handwriting from extended character-shapes may be easier, synthetic strokes add to the shape-variability of the synthesized handwriting.

As in natural data, the improvement due to the injection of synthesized data may gradually reach saturation. In our case, six versions per each of the 721 Tunisian town/village names that we synthesized were enough for saturation. The extended glyphs technique resulted in an improvement of 9.93% and that of synthetic connections reached an improvement of 16.39% over the baseline system.

This work can be extended in a number of ways. Certain ligatures may be used instead of their corresponding unligative character-shapes. Generation-based synthesis can be used to increase the variability of character-shapes themselves. Other datasets can be used to enrich the investigations on their impact on different segmentation and recognition systems. Writing styles of specific writers can be captured and synthesized,

and their results can be tested by the accuracy of writer-identification systems in distinguishing them.

REFERENCES

[1] "Analysis," *Wikipedia, the free encyclopedia*. 26-Apr-2014.

[2] "Synthesis," *Wikipedia, the free encyclopedia*. 05-Apr-2014.

[3] P. Rao, "Shape vectors: An efficient parametric representation for the synthesis and recognition of hand script characters," *Sadhana*, vol. 18, no. 1, pp. 1–15, Mar. 1993.

[4] I. Guyon, "Handwriting Synthesis From Handwritten Glyphs," in *Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition*, 1996, pp. 309–312.

[5] M. Mori, A. Suzuki, A. Shio, and S. Ohtsuka, "Generating New Samples from Handwritten Numerals Based on Point Correspondence," in *7th Int. Workshop on Frontiers in Handwriting Recognition*, 2000, pp. 281–290.

[6] T. M. Ha and H. Bunke, "Off-Line Handwritten Numeral String Recognition," *Pattern Recognit.*, vol. 31, pp. 257–272, 1997.

[7] H. S. Baird, "The State of the Art of Document Image Degradation Modeling," in *Proc. of 4th IAPR International Workshop on Document Analysis Systems, Rio de Janeiro*, 2000, pp. 1–16.

[8] J. Cano, J.-C. Perez-Cortes, J. Arlandis, and R. Llobet, "Training Set Expansion in Handwritten Character Recognition," in *Structural, Syntactic and Statistical Pattern Recognition, pages 548– 556. LNCS 2396*, 2002, pp. 548–556.

[9] I. Ahmad, L. Rothacker, G. A. Fink, and S. A. Mahmoud, "Novel Sub-character HMM Models for Arabic Text Recognition," in *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 658–662.

[10] M. Helmers and H. Bunke, “Generation and Use of Synthetic Training Data in Cursive Handwriting Recognition,” in *Pattern Recognition and Image Analysis*, F. J. Perales, A. J. C. Campilho, N. P. de la Blanca, and A. Sanfeliu, Eds. Springer Berlin Heidelberg, 2003, pp. 336–345.

[11] T. Varga and H. Bunke, “Perturbation Models for Generating Synthetic Training Data in Handwriting Recognition,” in *Machine Learning in Document Analysis and Recognition*, P. S. Marinai and P. H. Fujisawa, Eds. Springer Berlin Heidelberg, 2008, pp. 333–360.

[12] H. Miyao and M. Maruyama, “Virtual Example Synthesis Based on PCA for Off-line Handwritten Character Recognition,” in *Proceedings of the 7th International Conference on Document Analysis Systems*, Berlin, Heidelberg, 2006, pp. 96–105.

[13] Z. Lin, L. Wan, C.-H. Hu, and J. Wang, “Handwriting recognition training and synthesis,” US7657094 B202-Feb-2010.

[14] R. M. Saabni and J. A. El-Sana, “Comprehensive synthetic Arabic database for on/off-line script recognition research,” *Int. J. Doc. Anal. Recognit. IJDAR*, vol. 16, no. 3, pp. 285–294, Sep. 2013.

[15] SRIHARI, S. N., SRINIVASAN, H., HUANG, C., and SHETTY, S, “Spotting words in Latin, Devanagari and Arabic scripts,” *Vivek Indian J. Artif. Intell.*, vol. 16, no. 3, pp. 2–9, 2006.

[16] S. Madhvanath, V. Govindaraju, and S. Member, “The Role of Holistic Paradigms in Handwritten Word Recognition,” *IEEE Trans PAMI*, pp. 149–164.

[17] Z. Lin and L. Wan, “Style-preserving English handwriting synthesis,” *Pattern Recognit.*, vol. 40, no. 7, pp. 2097–2109, Jul. 2007.

[18] A. O. Thomas, A. Rusu, and V. Govindaraju, “Synthetic Handwritten CAPTCHAs,” *Pattern Recogn*, vol. 42, no. 12, pp. 3365–3373, Dec. 2009.

[19] J. Dolinsky and H. Takagi, “Synthesizing Handwritten Characters Using Naturalness Learning,” in *IEEE International Conference on Computational Cybernetics, 2007. ICCC 2007*, 2007, pp. 101–106.

[20] “VISTAWIDE: World Languages and Cultures,” *The Arabic Language*. [Online]. Available: http://www.vistawide.com/arabic/arabic.htm. [Accessed: 04-Jan-2014].

[21] D. DeSilver, “World’s Muslim population more widespread than you might think,” *Pew Research Center*, 07-Jun-2013. .

[22] Ibrir, “ بنية الخطاب العلمي في كتاب سيبويه مخارج الحروف عينة,” *مجلة كلية الآداب والعلوم الإنسانية والاجتماعية*, vol. 7, Jun-2012.

[23] *Alphabet and Pronunciation. Babylon Academy*. .

[24] “Typographic ligature,” *Wikipedia, the free encyclopedia*. 05-Apr-2014.

[25] M. L. Elyaakoubi, “Justify Just or Just Justify,” *J. Electron. Publ.*, vol. 13, no. 1, Winter 2010.

[26] “مجمع الملك فهد لطباعة المصحف الشريف بالمدينة المنورة.” [Online]. Available: http://www.qurancomplex.org/?Lan=ar. [Accessed: 07-Apr-2014].

[27] J. T. Andrew, A. Gillies, E. Erl, S. Schlosser, and S. Cavin, *Advances In Arabic Text Recognition*. .

[28] J. Wang, C. Wu, Ying-Qing, Xu, Y. Xu, and H. Shum, *Combining Shape and Physical Models for On-Line Cursive Handwriting Synthesis*. 2003.

[29] W.-D. Chang and J. Shin, “A statistical handwriting model for style-preserving and variable character synthesis,” *Int. J. Doc. Anal. Recognit. IJDAR*, vol. 15, no. 1, pp. 1–19, Mar. 2012.

[30] G. Gangadhar, D. Joseph, and V. S. Chakravarthy, “An oscillatory neuromotor model of handwriting generation,” *Int. J. Doc. Anal. Recognit. IJDAR*, vol. 10, no. 2, pp. 69–84, Nov. 2007.

[31] J. Galbally, J. Fierrez, J. Ortega-Garcia, and R. Plamondon, “Synthetic on-line signature generation. Part II: Experimental validation,” *Pattern Recognit.*, vol. 45, no. 7, pp. 2622–2632, Jul. 2012.

[32] J. Galbally, R. Plamondon, J. Fierrez, and J. Ortega-Garcia, “Synthetic On-line Signature Generation. Part I: Methodology and Algorithms,” *Pattern Recogn*, vol. 45, no. 7, pp. 2610–2621, Jul. 2012.

[33] C. O’Reilly and R. Plamondon, “Development of a Sigma–Lognormal representation for on-line signatures,” *Pattern Recognit.*, vol. 42, no. 12, pp. 3324–3337, Dec. 2009.

[34] N. Vincent, A. Seropian, and G. Stamon, “Synthesis for handwriting analysis,” *Pattern Recognit. Lett.*, vol. 26, no. 3, pp. 267–275, Feb. 2005.

[35] S. Bayoudh, H. Mouchère, L. Miclet, and E. Anquetil, “Learning a Classifier with Very Few Examples: Analogy Based and Knowledge Based Generation of New Examples for Character Recognition,” in *Machine Learning: ECML 2007*, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Springer Berlin Heidelberg, 2007, pp. 527–534.

[36] L. JIN and X. N. Zu, “Synthesis of Chinese Character Using Affine Transformation,” in *Ninth International Conference on Document Analysis and Recognition, 2007. ICDAR 2007*, 2007, vol. 1, pp. 218–222.

[37] R. Saabni and J. El-Sana, “Efficient Generation of Comprehensive Database for Online Arabic Script Recognition,” in *10th International Conference on Document Analysis and Recognition, 2009. ICDAR ’09*, 2009, pp. 1231–1235.

[38] T. Varga and H. Bunke, “Generation of Synthetic Training Data for an HMM-based Handwriting Recognition System,” in *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, 2003, pp. 618–622.

[39] Y. Zheng and D. Doermann, “Handwriting matching and its application to handwriting synthesis,” in *Proceedings of ICDAR*, 2005, pp. 861–865.

[40] M. Djioua and R. Plamondon, “An interactive system for the automatic generation of huge handwriting databases from a few specimens,” in *19th International Conference on Pattern Recognition, 2008. ICPR 2008*, 2008, pp. 1–4.

[41] T. Varga and H. Bunke, “Off-line handwritten textline recognition using a mixture of natural and synthetic training data,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, 2004, vol. 2, pp. 545–549 Vol.2.

[42] P. Viswanath, M. N. Murty, and S. Bhatnagar, “A Pattern Synthesis Technique with an Efficient Nearest Neighbor Classifier for Binary Pattern Recognition,” in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 4 - Volume 04*, Washington, DC, USA, 2004, pp. 416–419.

[43] G. Zi and D. Doermann, “Document image ground truth generation from electronic text,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, 2004, vol. 2, pp. 663–666 Vol.2.

[44] J. Wang, C. Wu, Y.-Q. Xu, H.-Y. Shum, and L. Ji, “Learning-based cursive handwriting synthesis,” in *Eighth International Workshop on Frontiers in Handwriting Recognition, 2002. Proceedings*, 2002, pp. 157–162.

[45] H. Choi, S. J. Choi, and J.-H. Kim, “Writer dependent online handwriting generation with Bayesian network,” in *Ninth International Workshop on Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004*, 2004, pp. 130–135.

[46] C. V. Jawahar and A. Balasubramanian, “Synthesis of Online Handwriting in Indian Languages,” presented at the Tenth International Workshop on Frontiers in Handwriting Recognition, 2006.

[47] L. Ballard, D. Lopresti, and F. Monrose, “Evaluating the security of handwriting biometrics,” in *In The 10 th International Workshop on the Foundations of Handwriting Recognition*, 2006, pp. 461–466.

[48] M. Kokula, *Automatic Generation of Script Font Ligatures Based on Curve Smoothness Optimization*. 1994.

[49] M. Sarfraz and M. A. Khan, "Automatic outline capture of Arabic fonts," *Inf. Sci.*, vol. 140, no. 3–4, pp. 269–281, Feb. 2002.

[50] P. Liu, L. Ma, F. K.-P. Soong, and Y.-J. Wu, "Hidden markov model based handwriting/calligraphy generation," WO2009023648 A219-Feb-2009.

[51] J. A. R. Serrano and F. C. Perronnin, "Handwritten word spotter using synthesized typed queries," US20100067793 A118-Mar-2010.

[52] A. L. Coates, H. S. Baird, and R. J. Faternan, "Pessimal print: a reverse Turing test," in *Sixth International Conference on Document Analysis and Recognition, 2001. Proceedings*, 2001, pp. 1154–1158.

[53] Y. Elarian, "ARABIC TEXT STEGANOGRAPHY USING MULTIPLE DIACRITICS." [Online]. Available: http://www.academia.edu/4863456/ARABIC_TEXT_STEGANOGRAPHY_USING_MULTIPLE_DIACRITICS. [Accessed: 14-Feb-2014].

[54] A. Odeh, K. Elleithy, and M. Faezipour, "Steganography in Arabic text using Kashida variation algorithm (KVA)," in *Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island*, 2013, pp. 1–6.

[55] H. Choi, S.-J. Cho, and J. H. Kim, "Generation of Handwritten Characters with Bayesian Network Based On-line Handwriting Recognizers," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, Washington, DC, USA, 2003, p. 995.

[56] Y. Elarian, Husni Al-Muhtaseb, and Lahouari Ghouti, "Arabic Handwriting Synthesis," in *International Workshop on Frontiers in Arabic Handwriting Recognition*, Istanbul, 2010.

[57] E.-M. Nel, J. A. du Preez, and B. M. Herbst, “Verification of dynamic curves extracted from static handwritten scripts,” *Pattern Recognit.*, vol. 41, no. 12, pp. 3773–3785, Dec. 2008.

[58] H. Miyao, M. Maruyama, Y. Nakano, and T. Hananoi, “Off-line handwritten character recognition by SVM based on the virtual examples synthesized from on-line characters,” in *Eighth International Conference on Document Analysis and Recognition, 2005. Proceedings*, 2005, pp. 494–498 Vol. 1.

[59] H. Fujioka, H. Kano, H. Nakata, and H. Shinoda, “Constructing and reconstructing characters, words, and sentences by synthesizing writing motions,” *IEEE Trans. Syst. Man Cybern. Part Syst. Hum.*, vol. 36, no. 4, pp. 661–670, 2006.

[60] Y. Wang, H. Wang, C. Pan, and L. Fang, “Style preserving Chinese character synthesis based on hierarchical representation of character,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, 2008, pp. 1097–1100.

[61] V. Margner and M. Pechwitz, “Synthetic data for Arabic OCR system development,” in *Sixth International Conference on Document Analysis and Recognition, 2001. Proceedings*, 2001, pp. 1159–1163.

[62] “Solid_modeling,” *Wikipedia*, 2014. [Online]. Available: http://en.wikipedia.org/wiki/Solid_modeling.

[63] Jan Dolinsky, “Statistical Modelling,” *Statistical Modelling*, 2014. [Online]. Available: http://jandolinsky.com/modeling.html.

[64] R. Plamondon and M. Djioua, “A multi-level representation paradigm for handwriting stroke generation,” *Hum. Mov. Sci.*, vol. 25, no. 4–5, pp. 586–607, Oct. 2006.

[65] O. Stettiner and D. Chazan, “A statistical parametric model for recognition and synthesis of handwriting,” in *Proceedings of the 12th IAPR International*, 1994, vol. 2, pp. 34–38 vol.2.

[66] J. Dolinsky, *Naturalness Learning and Its Application to the Synthesis of Handwritten Characters(Soft Computing,<Special Issue>Doctorial Theses on Aritifical Intelligence)*, vol. 25. 2010.

[67] V. G. Achint Oommen Thomas, "Generation and Performance Evaluation of Synthetic Handwritten CAPTCHAs."

[68] L. Ballard, F. Monrose, and D. Lopresti, "Biometric Authentication Revisited: Understanding the Impact of Wolves in Sheep's Clothing," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, Berkeley, CA, USA, 2006.

[69] J. Chen, W. Cheng, and D. Lopresti, "Using perturbed handwriting to support writer identification in the presence of severe data constraints," 2011.

[70] Y.-Q. Xu, H.-Y. Shum, J. Wang, and C. Wu, "Learning-based system and process for synthesizing cursive handwriting," US7227993 B205-Jun-2007.

[71] P. Viswanath, N. Murty, and S. Bhatnagar, "Overlap pattern synthesis with an efficient nearest neighbor classifier," *Pattern Recognit.*, vol. 38, no. 8, pp. 1187–1195, Aug. 2005.

[72] P. Viswanath, M. N. Murty, and S. Bhatnagar, "Partition Based Pattern Synthesis Technique with Efficient Algorithms for Nearest Neighbor Classification," *Pattern Recogn Lett*, vol. 27, no. 14, pp. 1714–1724, Oct. 2006.

[73] W. Cheng and D. Lopresti, "Parameter calibration for synthesizing realistic-looking variability in offline handwriting," 2011, vol. 7874.

[74] C. Chelba, D. Bikel, M. Shugrina, P. Nguyen, and S. Kumar, "Large Scale Language Modeling in Automatic Speech Recognition," *CoRR*, vol. abs/1210.8440, 2012.

[75] H. Chun-Hui, L. Zhouchen, W. Liang, and W. Jian, "Handwriting recognition training and synthesis," WO2007079009 A112-Jul-2007.

[76] J. Dolinsky and H. Takagi, "Analysis and Modeling of Naturalness in Handwritten Characters," *IEEE Trans. Neural Netw.*, vol. 20, no. 10, pp. 1540–1553, Oct. 2009.

[77] H. Bezine, A. M. Alimi, and N. Derbel, “Handwriting trajectory movements controlled by a beta-elliptical model,” in *Proc. Seventh Int’l Conf. Document Analysis and Recognition*, 2003, pp. 1228–1232.

[78] J. M. Hollerbach, “An oscillation theory of handwriting,” *Biol. Cybern.*, vol. 39, no. 2, pp. 139–156, Jan. 1981.

[79] B. L. Elbert, “Analysis by synthesis in handwriting recognition,” Thesis, Massachusetts Institute of Technology, 1997.

[80] “Cursive,” *Wikipedia*. [Online]. Available: http://en.wikipedia.org/wiki/Cursive.

[81] M. Djioua and R. Plamondon, “Studying the variability of handwriting patterns using the Kinematic Theory,” *Hum. Mov. Sci.*, vol. 28, no. 5, pp. 588–601, Oct. 2009.

[82] R. Plamondon, “A kinematic theory of rapid human movements,” *Biol. Cybern.*, vol. 72, no. 4, pp. 309–320, Mar. 1995.

[83] R. Plamondon, “A kinematic theory of rapid human movements: Part III. Kinetic outcomes,” *Biol. Cybern.*, vol. 78, no. 2, pp. 133–145, Feb. 1998.

[84] R. Plamondon, X. Li, and M. Djioua, “Extraction of delta-lognormal parameters from handwriting strokes,” *Front. Comput. Sci. China*, vol. 1, no. 1, pp. 106–113, Feb. 2007.

[85] R. Plamondon, C. Feng, and A. Woch, “A kinematic theory of rapid human movement. Part IV: a formal mathematical proof and new insights,” *Biol. Cybern.*, vol. 89, no. 2, pp. 126–138, Aug. 2003.

[86] W. Guerfali and R. Plamondon, “The Delta LogNormal theory for the generation and modeling of cursive characters,” in , *Proceedings of the Third International Conference on Document Analysis and Recognition, 1995*, 1995, vol. 1, pp. 495–498 vol.1.

[87] M. Djioua and R. Plamondon, “A New Algorithm and System for the Characterization of Handwriting Strokes with Delta-Lognormal Parameters,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2060–2072, Nov. 2009.

[88] P. Simard and Y. L. Cun, “Reverse TDNN: An Architecture for Trajectory Generation,” in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 4*, 1991, pp. 579–588.

[89] M.A. Slim, A. Abdelkarim, and M. Benrejeb, “Handwriting Process Modelling by Artificial Neural Networks,” *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 5, pp. 297–307, 2013.

[90] A. Almaksour, E. Anquetil, R. Plamondon, and C. O’Reilly, “Synthetic Handwritten Gesture Generation Using Sigma-Lognormal Model for Evolving Handwriting Classifiers,” presented at the 15th Biennial Conference of the International Graphonomics Society (2011), pp. 98–101.

[91] A. Almaksour, E. Anquetil, S. Quiniou, and M. Cheriet, “Evolving Fuzzy Classifiers: Application to Incremental Learning of Handwritten Gesture Recognition Systems,” in *2010 20th International Conference on Pattern Recognition (ICPR)*, 2010, pp. 4056–4059.

[92] R. Plamondon, C. O’Reilly, J. Galbally, A. Almaksour, and É. Anquetil, “Recent developments in the study of rapid human movements with the kinematic theory: Applications to handwriting and signature synthesis,” *Pattern Recognit. Lett.*, vol. 35, pp. 225–235, Jan. 2014.

[93] T. Wakahara, Y. Kimura, and A. Tomono, “Affine-invariant recognition of gray-scale characters using global affine transformation correlation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 4, pp. 384–395, Apr. 2001.

[94] D. Keysers, C. Gollan, and H. Ney, “Local context in non-linear deformation models for handwritten character recognition,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, 2004, vol. 4, pp. 511–514 Vol.4.

[95] H. Bezine, A. M. Alimi, and N. Sherkat, “Generation and analysis of handwriting script with the beta-elliptic model,” in *Ninth International Workshop on Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004*, 2004, pp. 515–520.

[96] M. Ltaief, H. Bezine, and A. M. Alimi, “A Neuro-beta-Elliptic Model for Handwriting Generation Movements,” in *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012, pp. 803–808.

[97] L. Dinges, A. Al-Hamadi, and M. Elzobi, “An Approach for Arabic Handwriting Synthesis Based on Active Shape Models,” in *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 1260–1264.

[98] L. Dinges, M. Elzobi, A. Al-Hamadi, and Z. A. Aghbari, “Synthizing Handwritten Arabic Text Using Active Shape Models,” in *Image Processing and Communications Challenges 3*, R. S. Choraś, Ed. Springer Berlin Heidelberg, 2011, pp. 401–408.

[99] S. Mozaffari, K. Faez, V. MäRgner, and H. El Abed, “TWO-STAGE LEXICON REDUCTION FOR OFFLINE ARABIC HANDWRITTEN WORD RECOGNITION,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 22, no. 07, pp. 1323–1341, Nov. 2008.

[100] A. AbdelRaouf, C. A. Higgins, and M. Khalil, “A Database for Arabic Printed Character Recognition,” in *Image Analysis and Recognition*, A. Campilho and M. Kamel, Eds. Springer Berlin Heidelberg, 2008, pp. 567–578.

[101] Y. Elarian and F. Idris, “A Lexicon of Connected Components for Arabic Optical Text Recognition,” in *1st International Workshop on Frontiers in Arabic Handwriting Recognition (FAHR2010), in conjunction with the 20th International Conference on Pattern Recognition (ICPR)*, Istanbul, 2010.

[102] Y. Haralambous and A. F. Virus, “The traditional Arabic typecase extended to the Unicode set of glyphs,” *Electron. Publ. Dissem. Des.*, vol. 8, 1995.

[103] Y. Haralambous, “Simplification of the arabic script: Three different approaches and their implementations,” in *Electronic Publishing, Artistic Imaging, and Digital Typography*, R. D. Hersch, J. André, and H. Brown, Eds. Springer Berlin Heidelberg, 1998, pp. 138–156.

[104] F. Menasri, N. Vincent, E. Augustin, and M. Cheriet, “Shape-Based Alphabet for Off-line Arabic Handwriting Recognition,” in *Ninth International Conference on Document Analysis and Recognition, 2007. ICDAR 2007*, 2007, vol. 2, pp. 969–973.

[105] Y. Al-Ohali, M. Cheriet, and C. Suen, “Databases for recognition of handwritten Arabic cheques,” *Pattern Recognit.*, vol. 36, no. 1, pp. 111–121, Jan. 2003.

[106] S. A. M. Husni A Al-Muhtaseb, “A novel minimal script for Arabic text recognition databases and benchmarks,” 2009.

[107] V. Märgner and H. El Abed, “Databases and Competitions: Strategies to Improve Arabic Recognition Systems,” in *Proceedings of the 2006 Conference on Arabic and Chinese Handwriting Recognition*, Berlin, Heidelberg, 2008, pp. 82–103.

[108] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri, “IFN/ENIT - database of handwritten Arabic words,” in *In Proc. of CIFED 2002*, 2002, pp. 129–136.

[109] Hashim Mohammed al-Baghdadi, *rules of Arabic calligraphy*. 1961.

[110] Naser Abdelwahab Al-Nassary, *The Ruqaa Style Workbook: The best way to teach the Ruqaa calligraphic style* كراسة خط الرقعة أفضل طريقة لتعليم خط الرقعة.

[111] A. Gillies, E. Erlandson, J. Trenkle, and S. Schlosser, *Arabic Text Recognition System*. 1999.

[112] Aqil Azmi and Abeer Alsaiari, “Arabic Typography. A Survey,” *Int. J. Electr. Comput. Sci.*, vol. 9, no. 10, pp. 16–22, 2010.

[113] The Unicode Consortium, “Unicode.” [Online]. Available: http://www.unicode.org/charts/PDF/. [Accessed: 07-Apr-2014].

[114] Gyeonghwan Kim, Venu Govindaraju, and Sargur N Srihari, “An architecture for handwriting text recognition systems,” *Int J Doc. Anal. Recognit.*, vol. 2, pp. 37–44, 1999.

[115] D. Guillevic and C. Y. Suen, “Cursive script recognition applied to the processing of bank cheques,” in , *Proceedings of the Third International Conference on Document Analysis and Recognition, 1995*, 1995, vol. 1, pp. 11–14 vol.1.

[116] "Logology," *Wikipedia, the free encyclopedia*. 04-Apr-2014.

[117] "Pangram," *Wikipedia, the free encyclopedia*. 05-Apr-2014.

[118] "Lipogram," *Wikipedia, the free encyclopedia*. 05-Apr-2014.

[119] "07 *ويكيبيديا، الموسوعة الحرة*. [عدل]," جامع حروف, Apr-2014.

[120] "List of pangrams," *Wikipedia, the free encyclopedia*. 07-Apr-2014.

[121] M. H. Alsuwaiyel, *Algorithms: design techniques and analysis*. Singapore; New Jersey: World Scientific, 1999.

[122] "Set cover problem," *Wikipedia, the free encyclopedia*. 30-Mar-2014.

[123] Yahya Mir Allam, *The Contributions of Cryptologists in Arabic إسهامات علماء التعمية في اللسانيات العربية*, vol. 3. مكتبة ديوان العرب.

[124] S. Mamish and M. Cheriet, "Correcting Arabic OCR Errors Using Improved Topic-Based Language Models," *Int. J. Comput. Process. Lang.*, vol. 22, no. 04, pp. 321–340, Dec. 2009.

[125] B. Haddad and M. Yaseen, "Detection and Correction of Non-Words in Arabic: A Hybrid Approach," *Int. J. Comput. Process. Lang.*, vol. 20, no. 04, pp. 237–257, Dec. 2007.

[126] David Graff, "Arabic Gigaword Third Edition," *Linguistic Data Consortium*. [Online]. Available: http://catalog.ldc.upenn.edu/LDC2007T40. [Accessed: 18-Apr-2014].

[127] S. Maity, "Character Segmentation and Ground–Truth Preparation for Handwritten Bangla Word Images," Thesis, 2012.

[128] "UW-III English/Technical Document Image Database," University of Washington, Seattle, Washington, USA.

[129] Y. S. Elarian and S. A. Mahmoud, "An Adaptive Line Segmentation Algorithm (ALSA) for Arabic.," in *IPCV*, 2009, pp. 735–739.

[130] M. Thulke, V. Märgner, and A. Denge, "A General Approach to Quality Evaluation of Document Segmentation Results," in *Document Analysis Systems: Theory and Practice*, S.-W. Lee and Y. Nakano, Eds. Springer Berlin Heidelberg, 1999, pp. 43–57.

[131] F. Yin, Q.-F. Wang, and C.-L. Liu, “A Tool for Ground-Truthing Text Lines and Characters in Off-Line Handwritten Chinese Documents,” in *10th International Conference on Document Analysis and Recognition, 2009. ICDAR ’09*, 2009, pp. 951–955.

[132] Y. M. Alginahi, “A survey on Arabic character segmentation,” *Int. J. Doc. Anal. Recognit. IJDAR*, vol. 16, no. 2, pp. 105–126, Jun. 2013.

[133] J. J. Hull, “Document image skew detection: Survey and annotated bibliography,” in *Document Analysis Systems II. Word Scientific*, 1998, pp. 40–64.

[134] H. A. Al-Muhtaseb, S. A. Mahmoud, and R. S. Qahwaji, “Recognition of off-line printed Arabic text using Hidden Markov Models,” *Signal Process.*, vol. 88, no. 12, pp. 2902–2912, Dec. 2008.

[135] G. A. Abandah, “Printed and Handwritten Arabic Optical Character Recognition - Initial Study,” Supported by The Higher Council of Science and Technology, Amman, Jordan, Aug. 2004.

[136] M. Z. Khedher and G. Abandah, “Arabic Character Recognition using Approximate Stroke Sequence,” in *Arabic Language Resources and Evaluation*, 2002.

[137] “منتدى المخطـوطات والكتب النادرة - منتديات مكتبة المسجد النبوي الشريف.” [Online]. Available: http://www.mktaba.org/vb/forumdisplay.php?f=2. [Accessed: 30-Apr-2014].

[138] O. H. Schmitt, “A thermionic trigger,” *J. Sci. Instrum.*, vol. 15, no. 1, p. 24, Jan. 1938.

[139] H. Zhang, J. E. Fritts, and S. A. Goldman, “Image segmentation evaluation: A survey of unsupervised methods,” *Comput. Vis. Image Underst.*, vol. 110, no. 2, pp. 260–280, May 2008.

[140] M. Gong, Y. Liang, J. Shi, W. Ma, and J. Ma, “Fuzzy C-Means Clustering With Local Information and Kernel Metric for Image Segmentation,” *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 573–584, Feb. 2013.

[141] Yousef Elarian and Sameh Awaida, "Arabic Handwriting Dataset Samples," *Paragraph Database*. [Online]. Available: https://www.dropbox.com/sh/ad4opafx8f7isjw/0T376FTc-d. [Accessed: 23-Feb-2014].

[142] S. J. Young, G. Evermann, M. J. F. Gales, D. Kershaw, G. Moore, J. J. Odell, D. G. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, "The HTK book version 3.4," 2006.

[143] U.-V. Marti and H. Bunke, "Handwritten sentence recognition," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, 2000, pp. 463–466.

[144] R. Saabni and J. El-Sana, "Hierarchical On-line Arabic Handwriting Recognition," in *10th International Conference on Document Analysis and Recognition, 2009. ICDAR '09*, 2009, pp. 867–871.

Appendices

Appendix A: Statistics on character and ligature shapes sizes

Appendix A includes statistics on character-shape samples. Table 34 shows examples of statistics that can be taken from GTed data. The Width columns display the average and the standard deviation of the widths (in pixels) of the different character-shapes. This statistic is used in our non-blind segmentation of PAWs into words. The VP Height columns compute the maximum height in the VP profile of characters. This statistic can be used together with the widths statistics to design adaptive thresholds for alignment. We believe it provides more robust information than the mere height average.

Table 34. Statistics on the images of character-shape extracted from the UT and the IL PoDs scanned at 300 dpi.

SN	character -shape	Width (Pixels)		VP Height (Pixels)	
		Average	Standard Deviation	Average	Standard Deviation
1	بـ	23.66	7.36	22.06	6.85
2	لـ	23.39	7.78	40.44	11.03
3	غ	44.20	15.02	32.99	7.16
4	حـ	43.17	11.74	17.71	7.44
5	ا	20.01	7.00	32.40	12.08
6	ج	44.27	10.56	27.99	6.61
7	أ	23.76	7.77	34.33	11.27
8	ن	35.00	10.74	23.21	5.69
9	ا	17.09	7.58	34.50	12.21
10	خـ	39.54	11.09	21.91	8.18
11	ا	19.46	6.31	32.64	11.66
12	ه	27.57	10.46	20.59	4.90
13	ظ	44.27	8.77	38.06	10.77
14	مـ	42.36	19.79	25.03	13.03
15	آ	38.06	12.55	40.73	14.95
16	ن	39.14	14.75	22.26	5.54
17	بـ	22.47	7.05	20.04	6.82
18	و	35.33	9.34	23.73	4.90
19	ا	16.03	5.14	27.24	11.30
20	د	25.49	11.62	17.71	3.78
21	ي	49.00	14.61	26.91	6.75
22	ع	33.03	11.04	19.64	4.34
23	و	39.70	11.56	22.74	4.79
24	ف	59.80	23.83	23.16	5.59
25	ط	45.20	10.14	41.16	12.52
26	فـ	31.40	8.29	22.63	5.32
27	ق	55.04	14.96	27.40	6.42
28	يـ	32.54	12.39	22.09	5.77

29	ـسـ	45.64	10.89	14.80	4.58
30	ـعـ	30.93	7.87	19.84	5.15
31	ى	47.81	20.29	22.74	5.91
32	لا	46.40	13.56	39.96	12.02
33	ـحـ	42.03	11.81	18.36	8.53
34	ـضـ	45.86	11.13	24.79	6.23
35	ـا	17.87	5.41	30.60	12.21
36	ر	33.97	14.39	17.71	7.37
37	ـث	29.76	8.39	24.27	7.24
38	ـلا	48.17	13.11	41.09	9.53
39	ث	58.63	16.43	21.21	4.68
40	ق	34.44	9.02	22.80	5.20
41	ـر	38.26	10.25	19.53	6.85
42	ب	52.19	19.10	19.10	4.24
43	ز	36.23	11.73	21.50	6.48
44	ـمـ	30.33	9.17	16.24	3.36
45	ـز	39.76	9.61	22.49	6.58
46	م	39.79	14.68	33.14	11.74
47	ـت	27.81	6.86	23.66	6.16
48	ـنـ	28.21	10.94	17.21	5.00
49	ـجـ	38.38	9.20	20.33	6.89
50	ـبـ	34.93	8.78	16.45	4.71
51	ـه	46.72	24.59	22.98	8.23
52	ـمـ	28.57	8.13	16.21	3.59
53	ن	40.66	13.06	26.89	6.97
54	ـسـ	50.93	14.48	15.50	5.96
55	ـطـ	43.21	11.07	37.36	11.54
56	ـو	39.03	12.77	22.49	4.65
57	ع	52.80	24.33	32.29	9.49
58	و	39.85	12.77	21.80	5.79
59	ھ	41.57	10.31	26.97	6.28
60	ـبـ	34.89	10.68	16.72	4.77
61	چ	53.97	21.08	28.05	6.57
62	ا	14.81	5.02	31.85	13.81
63	ـلـ	18.54	5.64	34.00	13.19
64	ـشـ	50.02	16.44	20.44	4.91
65	ـمـ	35.63	11.30	19.06	5.37
66	س	73.69	30.08	26.81	6.12
67	ـحـ	45.87	13.91	19.17	8.87
68	ـك	69.71	25.60	21.46	6.00
69	ـع	33.93	11.99	19.16	3.97
70	ـو	42.13	14.07	22.07	4.80
71	ض	68.44	14.50	26.36	7.31
72	ا	14.33	5.51	32.82	10.83
73	ـلـ	17.56	6.31	35.51	10.97
74	ـشـ	51.85	16.35	21.65	5.25
75	ـبـ	37.09	11.90	17.58	5.24
76	خ	53.56	26.33	30.47	8.67
77	ـنـ	28.54	9.57	19.01	4.21
78	ـو	38.26	10.25	23.91	5.31
79	ح	49.67	17.43	27.66	8.31
80	ـب	35.07	14.78	20.36	5.59
81	ـصـ	52.37	13.49	22.29	4.98
82	ـد	31.63	12.20	18.11	5.87
83	د	26.24	7.59	18.03	3.86
84	ذ	28.53	8.39	18.89	4.84

85	ـلـ	16.40	4.75	35.63	10.79
86	ك	70.81	23.39	37.70	10.05
87	ف	28.73	6.51	21.91	5.51
88	أ	30.14	9.61	37.59	12.98
89	ـكـ	49.17	18.78	28.67	9.80
90	ر	39.16	12.93	21.67	6.99
91	ـمـ	31.19	12.49	17.24	4.47
92	ـه	43.20	22.81	24.53	8.75
93	و	42.86	16.25	20.67	4.31
94	صـ	51.54	11.00	21.61	6.82
95	ـب	60.09	20.76	18.80	4.98
96	و	38.96	11.29	20.89	5.46
97	تـ	32.87	9.65	19.74	7.01
98	ـكـ	58.44	27.64	32.99	12.58
99	ـلـ	28.53	14.60	38.51	10.62
100	ـفـ	65.00	27.33	24.67	6.79
101	و	39.81	11.43	22.16	5.30
102	قـ	34.21	9.01	22.40	6.79
103	ـا	19.41	7.11	34.90	11.53
104	ل	36.54	10.91	40.89	11.51
105	ـلـ	19.49	8.97	37.33	11.88
106	ـلآ	58.29	16.12	39.89	12.02
107	ت	55.57	18.36	20.64	5.47
108	أ	26.14	9.46	37.86	12.31
109	ع	36.06	11.95	19.89	4.63
110	ظـ	43.97	12.01	41.35	13.96
111	ـم	34.05	9.11	33.35	10.98
112	ضـ	54.77	12.66	23.29	5.61
113	ـبـ	33.44	11.43	14.94	3.97
114	ـط	55.54	20.12	37.16	11.97
115	سـ	51.83	16.40	15.34	5.10
116	ـهـ	48.19	25.09	27.33	7.69
117	ـيـ	32.31	9.63	17.17	4.57
118	ـل	47.51	17.29	44.27	12.99
119	و	40.94	11.61	22.24	5.36
120	أ	23.21	10.54	38.42	13.04
121	شـ	54.95	16.46	21.03	5.49
122	ـخـ	42.55	11.41	21.61	5.91
123	ـا	23.63	16.31	36.87	14.61
124	ص	67.90	13.60	26.34	5.82
125	ـلـ	31.34	18.67	35.97	11.45
126	ـص	78.09	18.66	26.43	5.74
127	ا	14.67	6.30	34.17	14.92
128	ـلـ	22.35	15.64	31.83	9.34
129	ـحـ	38.83	12.34	18.13	7.32
130	ي	61.15	34.88	25.85	5.93
131	غـ	38.27	11.80	22.26	4.27
132	ش	77.89	20.56	28.27	7.84
133	ر	37.86	12.23	20.00	8.67
134	ا	13.50	4.95	34.89	14.60
135	جـ	43.44	12.46	21.44	8.29
136	ـح	48.94	13.73	29.67	8.89
137	غـ	34.49	9.48	23.17	6.17
138	ـثـ	32.01	9.32	21.99	6.34
139	ـا	19.75	6.10	35.00	13.72
140	ـمـ	30.20	10.77	15.96	3.32

141	ـة	43.59	16.11	26.33	8.02
142	ـل	19.69	6.85	39.39	10.91
143	ـذ	30.24	7.19	21.53	5.38
144	ا	14.16	5.31	35.53	14.15
145	ـج	45.24	12.38	23.60	9.44
146	ـن	46.53	13.49	23.40	5.94
147	ـب	27.99	9.83	21.66	5.26
148	ـغ	33.81	8.66	24.70	6.36
149	ـي	39.11	10.47	18.06	5.29
150	ـظ	57.96	21.17	37.16	11.90
151	و	39.27	12.23	22.90	5.82
152	ا	14.91	5.78	33.39	11.17
153	ـن	25.01	9.75	20.99	5.45
154	ـق	40.50	12.03	24.70	4.89
155	ـض	73.50	18.42	29.07	7.15
156	ا	16.03	6.15	32.72	12.83
157	ـن	20.65	7.32	21.60	5.59
158	ـت	35.52	10.36	15.42	5.83
159	ـه	44.83	13.17	31.25	9.80
160	ـت	80.28	40.18	19.00	5.21
161	خ	46.16	16.7399	26.76	11.5155
162	س	59.76	31.4501	20.52	7.927379
163	ش	60.88	32.9689	24.2	11.39444
164	ط	63	29.3414	26.56	12.97459
165	ظ	58.76	27.3576	29.64	13.21262
166	غ	54.44	29.4067	34.84	12.89922
167	ق	53.04	25.5285	25.48	8.607748
168	ك	68.28	38.5957	31.08	11.48521

Appendix B: Statistics and comparisons on ligature shape widths

Table 35 displays the average widths of several UT PoD ligatures and compares them to the widths of the composing character-shape widths (from Appendix A) individually and when summed.

Table 35. Ligatures statistics extracted from GTed data scanned at 300 dpi.

Ligatures	Average width (Pixels)				
	Ligatures	1st character-shape	2nd character-shape	Sum of 1st & 2nd character-shapes	Ligatures - Sum
ظمـ	63.95	44.27	42.36	86.62	-22.67
ـنجـ	53.25	28.21	38.38	66.59	-13.34
جج	70.89	34.89	53.97	88.85	-17.96
ـشمـ	49.93	50.02	35.63	85.65	-35.72
جخ	67.60	37.09	53.56	90.65	-23.05
ظم	75.57	43.97	34.05	78.02	-2.45
سهـ	100.33	51.83	32.31	84.14	16.19
شخـ	74.25	54.95	42.55	97.50	-23.25
لحـ	31.22	22.35	38.83	61.19	-29.97
جح	63.50	43.44	48.94	92.38	-28.88
تهـ	67.50	35.52	44.83	80.35	-12.85

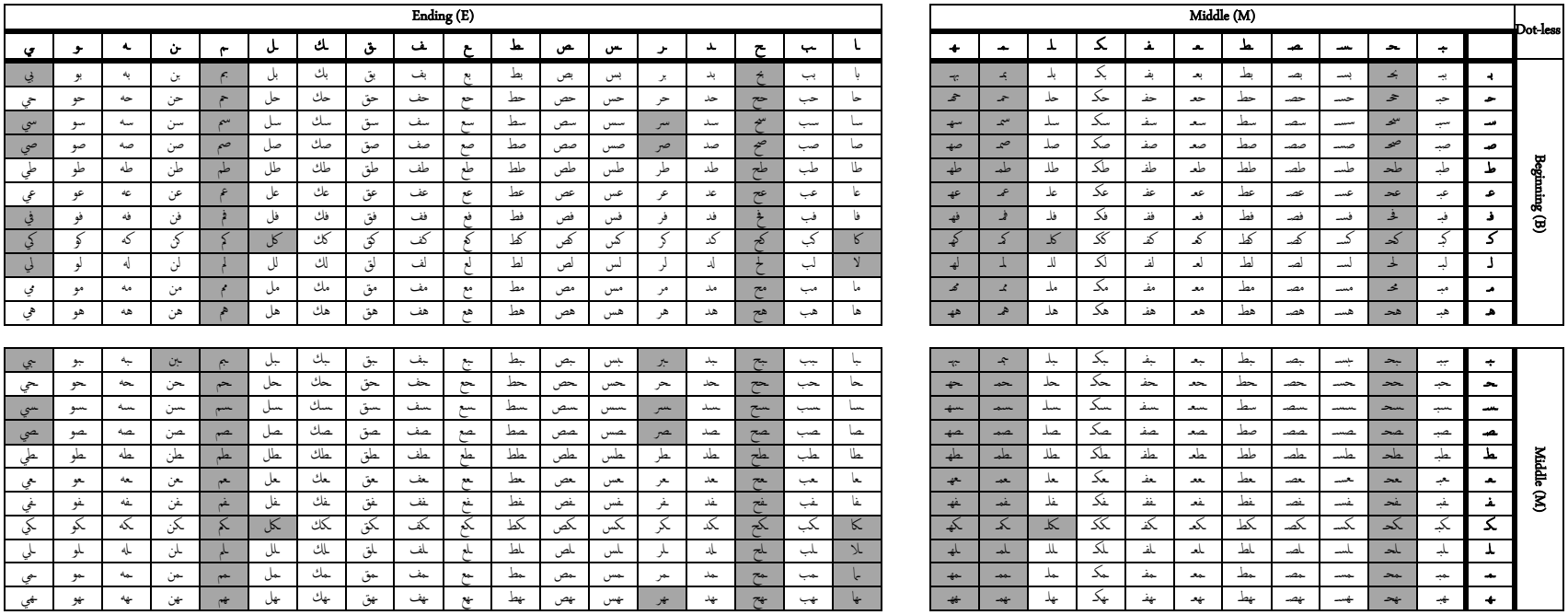
Appendix C: Number of samples per character-shape from our dataset

Table 36. Numbers of Samples per character-shape used in experiments

character-shape	Sample Count
ا	486
ـا	324
أ	162
ـأ	54
ـآ	54
ب	54
بـ	54
ـبـ	54
ـب	54
ت	54
تـ	54
ـتـ	48
ـت	108
ث	54
ثـ	54
ـثـ	54
ـث	54
ج	54
جـ	54
ـجـ	54
ـج	104
ح	54
حـ	54
ـحـ	54
ـح	162
خ	49
خـ	54
ـخـ	54
ـخ	54
د	108
ـد	54
ذ	54
ـذ	54
ر	108
ـر	108
ز	53
ـز	54
س	49
سـ	54
ـسـ	54
ـس	104
ش	49
شـ	54
ـشـ	102
ـش	50
ص	54
صـ	54
ـصـ	54
ـص	54
ض	54
ضـ	54
ـضـ	54
ـض	54
ط	49
طـ	54
ـطـ	54
ـط	54
ظ	53
ظـ	54
ـظـ	54
ـظ	38
ع	54
عـ	54
ـعـ	54
ـع	162
غ	49
غـ	54
ـغـ	54
ـغ	108
ف	54
فـ	54
ـفـ	54
ـف	54
ق	49
قـ	54
ـقـ	54
ـق	108
ك	49
كـ	54
ـكـ	54
ـك	54
ل	54
لـ	54
ـلـ	108
ـل	363
م	54
مـ	54
ـمـ	108
ـم	215
ن	108
نـ	54
ـنـ	50
ـن	162
ه	54
هـ	108
ـهـ	108
ـه	54
ة	54
و	324
ـو	270
ي	54
يـ	54
ـيـ	256
ـي	54
ى	54

Appendix D: Bigram shapes and ligatures

Table 37. Bigrams of the dot-less typographic model representing 548 out of all the possible 2,622.



Appendix E: Probabilities of the passage part.

The average difference between corresponding character probabilities in the passage part and Gigaword is approximately 0.13%.

character-shape	Gigaword	1000 Forms
ء	0.40%	0.41%
ا	10.97%	10.02%
ـا	7.01%	7.17%
ب	0.23%	0.21%
بـ	1.86%	1.63%
ـبـ	1.25%	1.30%
ـب	0.26%	0.35%
ت	0.96%	1.10%
تـ	1.36%	1.64%
ـتـ	2.31%	2.74%
ـت	0.38%	0.26%
ث	0.07%	0.07%
ثـ	0.20%	0.17%
ـثـ	0.25%	0.30%
ـث	0.07%	0.12%
ج	0.06%	0.06%
جـ	0.73%	0.66%
ـجـ	0.39%	0.46%
ـج	0.02%	0.06%
ح	0.09%	0.09%
حـ	0.70%	0.73%
ـحـ	0.58%	0.61%
ـح	0.06%	0.07%
خ	0.01%	0.00%
خـ	0.40%	0.47%
ـخـ	0.18%	0.29%
ـخ	0.02%	0.01%
د	1.21%	1.06%
ـد	2.15%	2.21%
ذ	0.17%	0.20%
ـذ	0.41%	0.59%
ر	1.92%	1.70%
ـر	3.48%	2.93%
ز	0.34%	0.34%
ـز	0.43%	0.34%
س	0.11%	0.09%
سـ	1.22%	1.09%
ـسـ	1.19%	1.04%
ـس	0.31%	0.34%
ش	0.02%	0.00%

character-shape	Gigaword	1000 Forms
شـ	0.47%	0.29%
ـشـ	0.65%	0.74%
ـش	0.04%	0.02%
ص	0.03%	0.05%
صـ	0.38%	0.34%
ـصـ	0.70%	0.58%
ـص	0.03%	0.05%
ض	0.07%	0.08%
ضـ	0.33%	0.33%
ـضـ	0.26%	0.29%
ـض	0.05%	0.06%
ط	0.03%	0.04%
طـ	0.34%	0.32%
ـطـ	0.63%	0.66%
ـط	0.09%	0.09%
ظ	0.01%	0.01%
ظـ	0.04%	0.04%
ـظـ	0.13%	0.20%
ـظ	0.01%	0.01%
ع	0.18%	0.19%
عـ	1.45%	1.64%
ـعـ	1.41%	1.67%
ـع	0.36%	0.43%
غ	0.00%	0.01%
غـ	0.17%	0.12%
ـغـ	0.21%	0.14%
ـغ	0.10%	0.02%
ف	0.16%	0.19%
فـ	0.58%	0.63%
ـفـ	0.63%	0.59%
ـف	0.16%	0.16%
ق	0.20%	0.18%
قـ	1.04%	0.97%
ـقـ	1.17%	1.25%
ـق	0.20%	0.22%
ك	0.09%	0.08%
كـ	0.90%	0.88%
ـكـ	0.75%	0.80%
ـك	0.11%	0.17%
ل	0.63%	0.61%

character-shape	Gigaword	1000 Forms
لـ	5.69%	5.90%
ـلـ	1.71%	1.89%
ـل	0.63%	0.79%
م	0.61%	0.46%
مـ	2.69%	2.58%
ـمـ	1.33%	1.55%
ـم	0.42%	0.55%
ن	1.21%	0.97%
نـ	1.32%	1.02%
ـنـ	1.66%	1.57%
ـن	1.54%	1.44%
ه	0.16%	0.24%
هـ	0.59%	0.77%
ـهـ	0.86%	1.11%
ـه	0.41%	0.51%
و	3.08%	3.44%
ـو	3.11%	2.58%
ي	0.08%	0.34%
يـ	2.38%	2.02%
ـيـ	4.36%	3.55%
ـي	0.23%	0.92%
ئ	0.00%	0.00%
ئـ	0.52%	0.41%
ـئـ	0.09%	0.12%
ـئ	0.00%	0.00%
ة	0.77%	0.84%
ـة	3.10%	3.33%
ى	0.42%	0.16%
ـى	1.57%	0.79%
ؤ	0.01%	0.01%
ـؤ	0.14%	0.19%
أ	0.48%	1.05%
ـأ	0.11%	0.26%
إ	0.15%	0.47%
ـإ	0.01%	0.04%
آ	0.03%	0.04%
ـآ	0.00%	0.00%

Vitae

Name	:Yousef S. I. Elarian
Nationality	:Palestine
Date of Birth	:9/11/1980
Email	:yousef.elarian@gmail.com

Yousef Elarian has submitted this dissertation to fulfil the requirements of his PhD degree in Computer Science and Engineering from King Fahd University of Petroleum & Minerals. He had received his MSc in Computer Engineering from the Jordanian University of Science and Technology, Jordan, in 2006 and his BSc in Computer Engineering from the Islamic University of Gaza in 2003 both with honors. His research interest is in Arabic computing, where he has several publications some of which are listed in this document.

Attended several workshops organized by the Deanship of Scientific Research, KFUPM, such as the Team-Based Learning, Implementing Social Media into Teaching and Learning, Formal Cooperative Learning-Design, Implementation and Assessment, Participatory Classrooms. Also attended the Academic Writing and Public Speaking, by the AMIDEAST, Gaza and the WebCT and Blackboard Online Teaching System, by IUG.

Applied for the Structured Encyclopediaproject through the Developing Innovation & Creativity Program at Saudi Universities, and the Arabic Handwriting: Analysis and Synthesis and got funded by KACST. Worked in the Automatic Arabic Optical Text Recognition (AOTR) and the Toward Content-Based Indexing and Retrieval of Arabic Manuscripts KFUPM funded Projects.

Prize Winner in the 1st National Scientific Conference for Higher Education Students, of the Third Graduate Day Seminar in KFUPM and in several other presentations.